LOAN DOCUMENT

PHOTOGRAPH THIS SHEET

DTIC ACCESSION NUMBER

LEVEL

INVENTORY

DOCUMENT IDENTIFICATION

DISTRIBUTION STATEMENT

HANDLE WITH CARE

ACCESSION FOR
NTIS GRA&I
DTIC TRAC
UNANNOUNCED
JUSTIFICATION

BY

DISTRIBUTION/

AVAILABILITY CODES

DISTRIBUTION	AVAILABILITY AND/OR SPECIAL	
A-1		

DISTRIBUTION STAMP

DATE ACCESSIONED

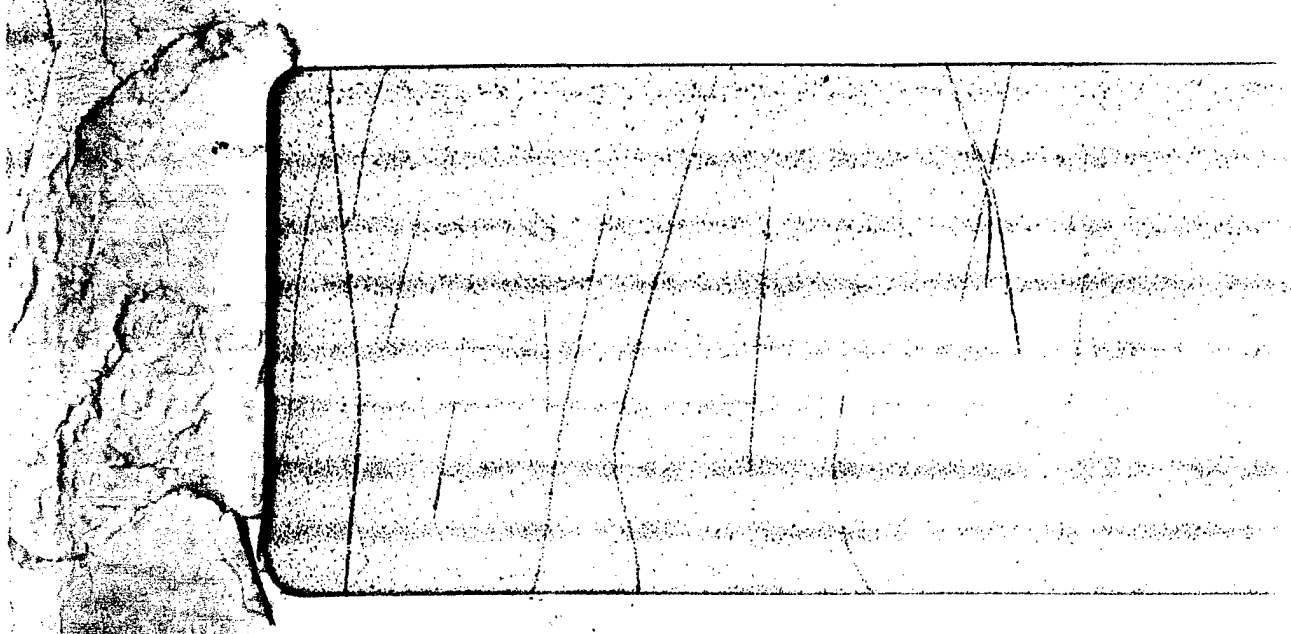
DATE RETURNED

19981223 015

DATE RECEIVED IN DTIC

REGISTERED OR CERTIFIED NUMBER

PHOTOGRAPH THIS SHEET AND RETURN TO DTIC-FDAC



AVIONICS DIVISION ITT
390 Washington Avenue, Nutley, N.J. 0711

REPORT NUMBER: D11803

REED-SOLOMON ENCODER/DECODER

COMPUTER PROGRAM

DESIGN SPECIFICATION

FEBRUARY 1976

PREPARED UNDER CONTRACT N62269-75-C-0503

DATA ITEM A005

FOR:

DEPARTMENT OF THE NAVY
NAVAL AIR DEVELOPMENT CENTER

BY:

ITT AVIONICS DIVISION
500 WASHINGTON AVENUE
NUTLEY, N. J. 07110

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Reed Solomon Encoder/Decoder Final Engineering Report COMPUTER PROGRAM DESIGN SPECIFICATION		5. TYPE OF REPORT & PERIOD COVERED FINAL ENGINEERING REPORT 18 JUN 75 - 9 FEB 76 6. PERFORMING ORG. REPORT NUMBER D 11803
7. AUTHOR(s) STAFF OF: CNI Systems Engineering		8. CONTRACT OR GRANT NUMBER(s) N62269-75-C-0503
9. PERFORMING ORGANIZATION NAME AND ADDRESS ITT Avionics Division 500 Washington Avenue Nutley, New Jersey 07110		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Design, Fab & Test RSED
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Air Development Center Warminster, Pennsylvania 18974		12. REPORT DATE February 9, 1976 13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Reed-Solomon
Encoder
Decoder
Round Trip Timing
Microprocessor
Galois Field
Errata

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The purpose of this program is to build a Reed-Solomon encoder/decoder capable of a 384 word/second (approx. 57.6 KBPS) throughput rate in a cost effective and practical manner.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

FIGURES

TABLES

TABLES

1.0 SCOPE

The title of the RSED microprocessor program is RSED1. This specification contains the detailed design description of RSED1 which is based upon the requirements defined in the document:
REED-SOLOMON ENCODER/DECODER Microprocessor Program Performance Specification, February, 1976, Data Item A004.
The programming implementation of the RSED functions performed by the microprocessor program will be presented in detail.The major functions are as follows:

- Generation of the erasure polynomial from the erasure locations.
- Modified syndrome computation from the product of the syndromes and the coefficients of the erasure polynomial.
- Generation of the error location polynomial derived from the modified syndromes.
- Solving for the roots of the error polynomial to obtain the locations of the errors.
- Generation ofthe errata polynomial based on the syndromes, the erasure polynomial and the error polynomial.
- Calculation of errata correction values.
- Correction of the data characters in error.
- Recognition of R/S code words without errata.
- Recognition of the following conditions which result in decode failures:

  1. $E > 16$ — E= number of erasures
  2. $2e + E > 16$ — e= number of errors
  3. The number of distinct roots of the error polynomial is less than the degree of the polynomial.

- Provision of the data quality of each word indicating if it was decoded and the number of errors found.
- Performance of these functions in minimum time.

2.0 REFERENCES

1. ITT Avionics
   R/S Encoder/Decoder Microprocessor Performance Specification,
   Data Item A004, Contract N62269-75-C-0503, NADC, 1976

2. ITT Avionics
   Requirements for RSED Microprocessor Program Documentation,
   Response to RFQ No. N62269-75-C-0503,NADC, 1975

3. American National Standard, Flowchart Symbols,
   ANSI X3.5-1970

4. 3001 Microprogram Control Unit
   Technical Memo Number MCS-268-0275/27.5k
   INTEL Corporation, Santa Clara, California

5. 3002 Central Processing Element
   Technical Memo Number MCS-269-0275/27.5k
   INTEL Corporation, Santa Clara, California

6. Series 3000 Cross Microprogramming System, CROMIS,
   Reference Specification, Revision A, MCS-487-0275/5k
   INTEL Corporation, Santa Clara, California

7. ITT Avionics
   Final Engineering Report, Reed-Solomon Encoder/Decoder,
   Data Item A003, Contract N62269-75-C-0503, NADC,1975

8. Berlekamp, Elwyn R. ,
   Algebraic Coding Theory
   New York: McGraw Hill, 1968

9. Lin, Shu
   An Introduction to Error Correcting Codes
   Englewood Cliffs, New Jersey: Prentice Hall, 1970

10. Berlekamp, Elwyn R. , ed.
    Key Papers in the development of Coding Theory
    New York: IEEE Press, 1972

11. Peterson, W. W. and Weldon, E. J. , Jr.
    Error Correcting Codes, 2nd Ed.
    Cambridge, Mass: MIT Press, 1972

3.0 REQUIREMENTS

3.1 Function Allocation

3.1.1 Processing Time Optimization

The overall function of RSED1 is to complete the decoding process (initiated by the RSED syndrome generator) minimizing the processing time. As described in the RSED Microprocessor Program Performance Specification, Data Item A004,Contract N62269-75-C-0503,NADC, 1975 , four R/S code words may be given to the RSED at the start of each time slot and the decoding results are required at the end of the same time slot. The four R/S code words consist of one short (16,4) word and three long (31,15) words. The two microprocessors utilized within the RSED each control their own input to the extent that they indicate when they are ready to accept input data. Reference RSED Microprocessor Program Performance Specification, section 3.1.2 . Each microprocessor will receive at least one word to process in a slot. A second R/S code word will be sent to a microprocessor only when it makes the request for another word to process. The time required to decode an R/S code word is variable depending upon the number and position of errors and erasures in the code word and upon the length of the code word. As part of the minimization of processing time , the three long code words are presented to the microprocessors ahead of the short word which is always sent as the fourth word to the available microprocessor. This ensures that the microprocessor which completes the processing of its first word before the other microprocessor, will be given the remaining long (the third) code word to process.

3.1.2 Design Structure

RSED1 is designed as a set of modular subprograms sharing a common data base. The processing sequence is fixed with the data content itself selecting the different processing paths taken.

RSED1 is utilized in a cyclical manner. There is a requirement that it should complete its processing within every time slot and be ready to accept new data in the next time slot. No microprocessor processing time is devoted to slot time checking, a hardware cycle reset pulse being provided at the start of each new time slot. This cycle reset forces a reinitialization of the microprocessor and its program and thus automatically terminates processing of data in the case of processing time overflow. The microprocessor is ready to accept new data as soon as its initialization program has run. The run time of the initialization program is minimal and is less than the syndrome generation time which is required by associated RSED hardware at the beginning of each time slot.

Whenever a Write Request for a new input word has been generated by RSED1 and after it has completed the remaining processing of its previous R/S code word, it enters a delay cycle until a new input code word has been transferred into its Data Ram.
Reference RSED Microprocessor Program Performance Specification, section 3.1.2 .

Two microprocessors are included in the RSED. They run in parallel utilizing the same program RSED1 but have no interface with each other. Refer to Figures 3.1, 3.2 and 3.3 .

3.1.3 Subprogram Functional Definition

The allocation of functions to the subprograms is based upon the computational process to be performed and upon the need to optimize the processing time. This optimization is twofold in that each separate function is itself optimized with respect to execution time and the sequence of functions is arranged whenever possible, to assist in the optimization of the longest processing paths. This has two effects. The first is that some short processing paths are deliberately not optimized whenever their optimization would compromise that of a longer processing path. The second effect is to assign functions to computational type subprograms which utilize the same data as a required housekeeping function. This is done to save memory access time etc. Table 3.1 summarizes the functions assigned to each subprogram.

3.2 Microprocessor Program Functional Flow Diagram

Figure 3.2 shows the data flow and control which exists between the microprocessors and the interfacing Data Steering and Control Unit. Figure 3.2 also illustrates the data flow between each microprocessor and its own Data Ram.

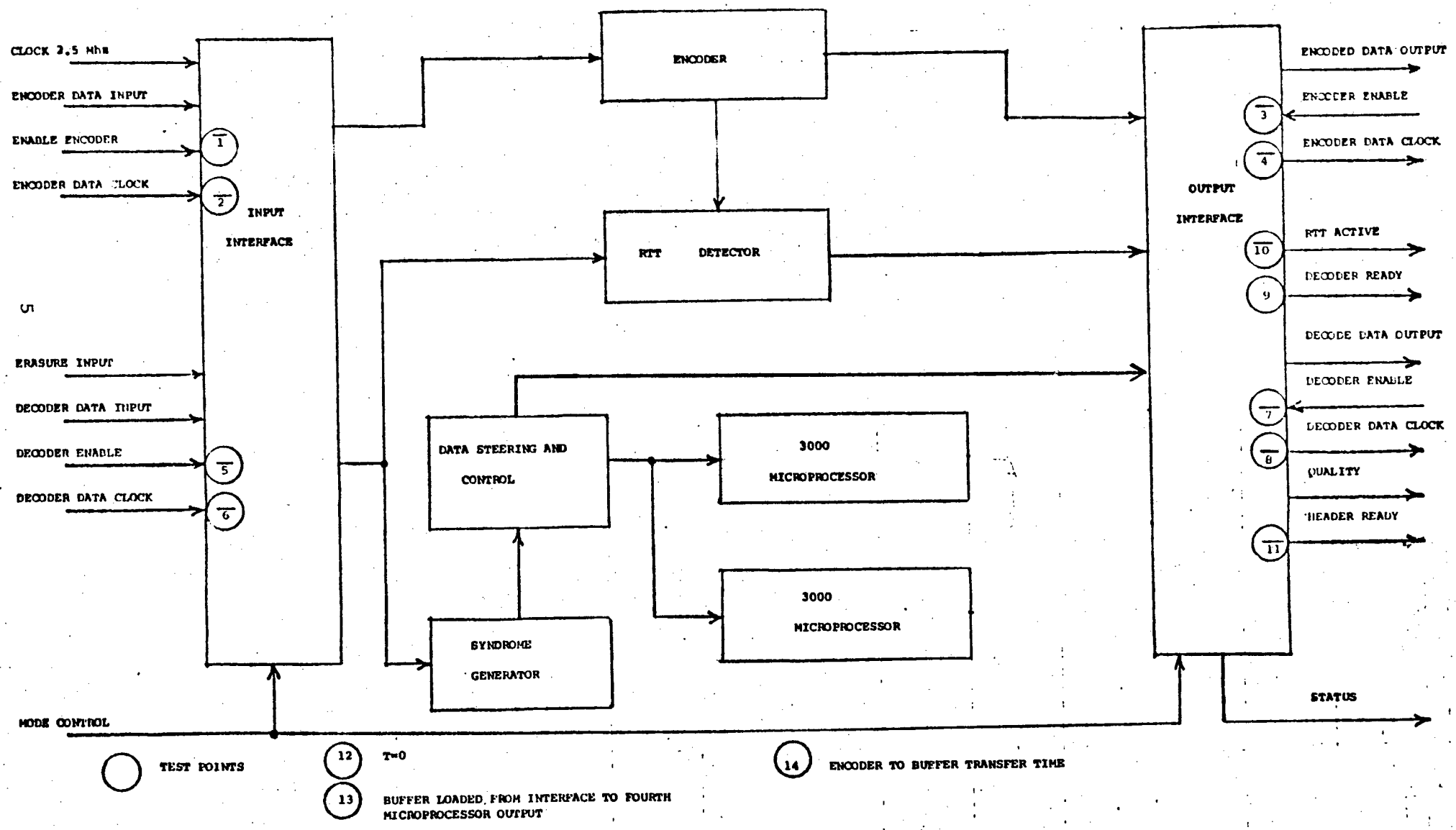
Figure 3.3 illustrates the sequence of control and the flow of data within each microprocessor.The three main areas of control are:

1. Initialization
   The master and cycle reset pulses are the stimuli which give control to the initializing subprogram.

2. Input/Output Control
   When processing is not being performed by the microprocessor, control is given to the input/output subprogram which cycles in a delay loop which checks for the receipt of new input data. As a time saving device the actual setting of the input/ output flags is shared between the input/output subprogram, the initializing subprogram and the computational subprograms. Reference section 4.0 for actual details.

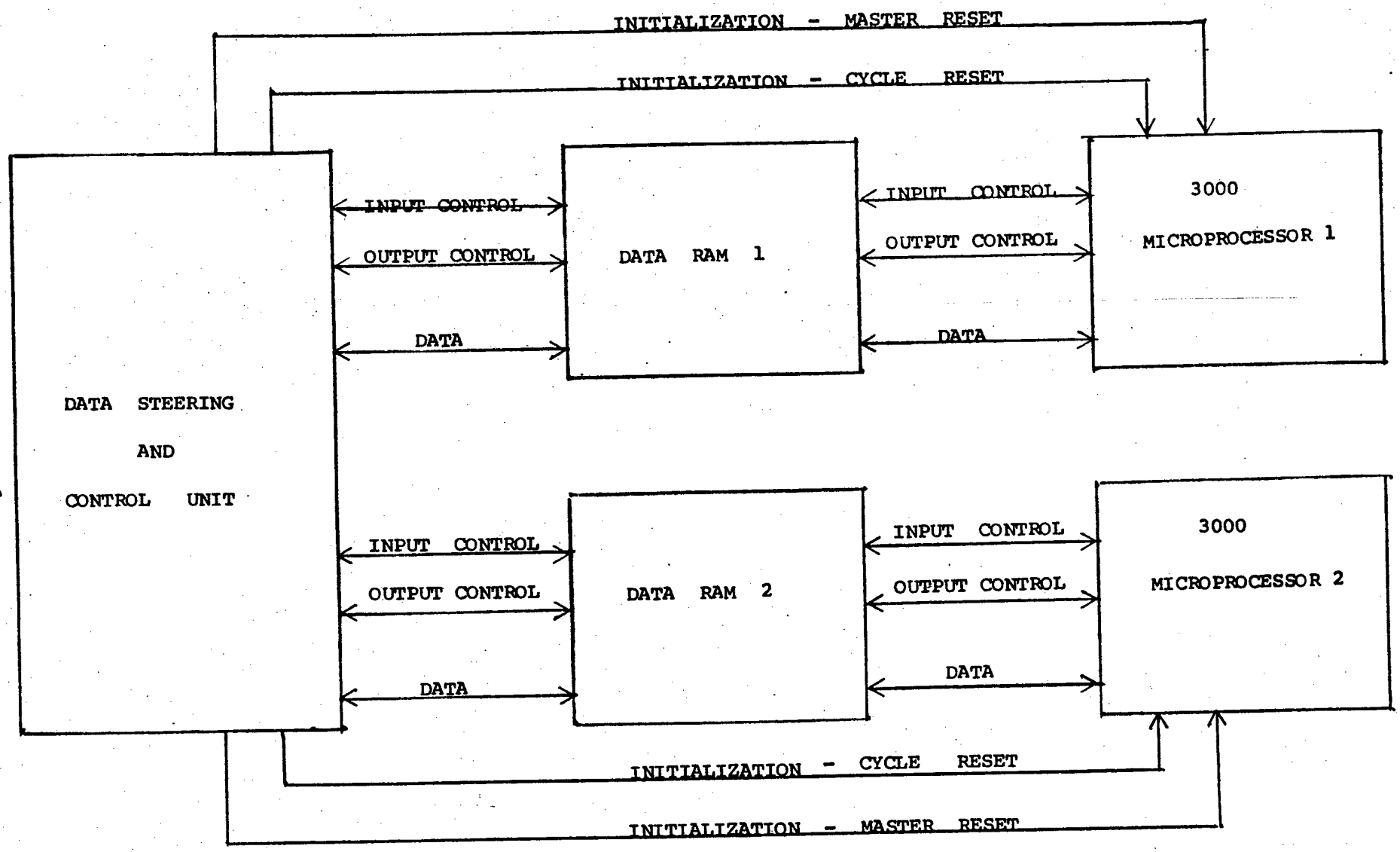
3. Processing control
   This control passes from one subprogram to the next as shown in Figure 3.3 , the data itself selecting the appropriate path. Control is passed to the input/output subprogram when processing is completed.

Data is passed between the subprograms either by depositing it in the common Data Ram or by utilizing a set of twelve internal registers within the microprocessor central processing element, CPE, as a time saving device .



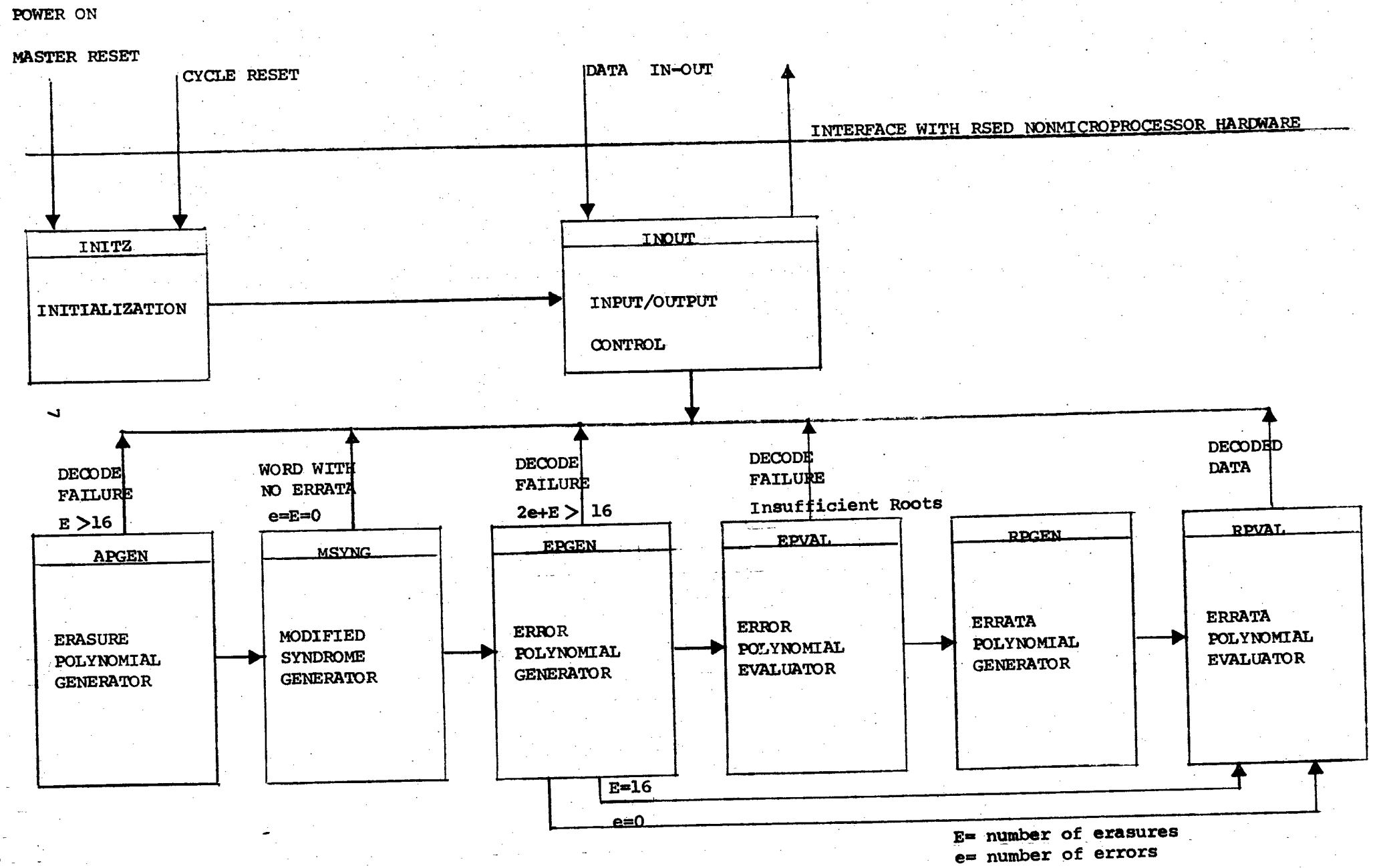
RSED BLOCK DIAGRAM

FIGURE 3.1



MICROPROCESSOR INTERFACE BLOCK DIAGRAM

FIGURE 3.2



RSED MICROPROCESSOR PROGRAM STRUCTURE -- CONTROL AND DATA FLOW

FIGURE 3.3

SUBPROGRAM FUNCTIONAL ASSIGNMENTS

SUBPROGRAM	FUNCTIONS	INPUTS	OUTPUTS
INITZ	1.Initialize Read/Write control flags.		IADXX/IBDXX IADZY/IBDZY
	2.Set current data block indicator.		R_9
	3.Transfer code/power conversion tables from ROM to RAM	Table CCTPC Table CPTCP	Table TCTPC Table TPTCC
INOUT	1.Control subprograms by providing an idle loop for microprocessor.		
	2.Perform cyclic initializations required.		IAD07/IBD07, TEPOL SIRCS/SIRPS, TEPDR
	3.Reset WRITE control flag on receipt of new R/S word.	IADZY IBDZY	IADZY IBDZY
	4.Set READ control flag to DATA READY when processing is complete.		IADXX IBDXX
	5.Switch data blocks for next R/S word.	R_9 SCBIX	R_9
APGEN	1.Store current data block indicator in Data Ram.	R_9	SCBIX
	2.Declares a decode failure if E>16.	IADES IBDES	
	3.Computes erasure polynomial coefficients.	IADEE IBDEE	TAPOL
MSYNG	1.Compute Modified syndromes and first s terms of partial errata polynomial.	TAPOL IADSN/IBDSN IADES/IBDES	TMSYN,TMSYP
	2.Determine if R/S code word has no errata.	IADSN/IBDSN IADES/IBDES	IAD07/IBD07

TABLE 3.1 Page 1/2

SUBPROGRAM FUNCTIONAL ASSIGNMENTS

SUBPROGRAM	FUNCTIONS	INPUTS	OUTPUTS
EPGEN	1.Compute cefficients of error polynomial. 2.Declare a decode failure if 2e + E 16. 3.Detect if e=0 or E=16 and if so bypass normal error processing.	TMSYN TMSYP	TEPOL
EPVAL	1.Determine the inverse roots of the error polynomial. 2.Declare a decode failure if the number of roots found is less than the degree of the error polynomial. 3.Stores the address of the error polynomial (a variable).	IADCS/IBDCS IADCC/IBDCC IADEE/IBDEE IADES/IBDES SCBIX TEPOL	TEVAL SEPEP
RPGEN	1.Completes the computation of the errata polynomial.	TMSYN TMSYP TEPOL SEPEP	TMYSN
RPVAL	1.Computes the errata correction values. 2.Corrects the data characters. 3.Sets the WRITE REQUEST flag to obtain a new R/S code word. 4.Report the result of the decoding process in the Data Quality word.	IADCS/IBDCS IADEE/IBDEE IADES/IBDES TEVAL TMSYN SEPEP	 IADCS/IBDCS IADZY/IBDZY IAD07/IBD07

TABLE 3.1 Page 2/2

3.3 Storage and Processing Allocation

3.3.1 Memory Storage

Table 3.2 shows the allocation of the Data Ram storage. The input/output storage areas are in fixed locations required by associated RSED hardware units. The remaining memory is allocated either to the common data base or to private tables used internally by individual subroutines. Each Data Ram is 1K by 8 bits .

The RSED1 program is stored in 1K by 36 bits PROMs.
Reference Appendix A,Specification for RSED Microprocessor RS-16 .
Allocation of subprograms to memory followed the requirements of the INTEL 3000 microprocessor which utilizes non-sequential programming steps as opposed to the normal sequential program counter method. Each instruction in the 3000 Series contains a jump function which is the next instruction to be selected. This provides a faster instruction execution time. In addition program branch points may only jump into certain specified memory cells. Reference INTEL 3001 and 3002 Technical Memos numbers MCS-268-0275/27.5k and MCS-269-0275/27.5k .

The 3000 microprocessor program memory is organised on a row , column basis. Each instruction is accessed in memory for execution by addressing it by its row, column address. Figure 3.4 illustrates this organization.

Figure 3.5 shows the main memory areas allocated to each subprogram and the common program branch point areas shared by all subprograms.

3.3.2 Processing Time

A processing time allocation is not allocated to the subprograms separately. The subprograms are an integrated set of programs which optimize processing time first by the allocation of functions to the individual subprogram so all possible processing both computational and housekeeping, is performed on data brought into the microprocessors CPE (central processing element) and secondly by a coding optimization within each subprogram.
Reference ITT Avionics Final Engineering Report, Data Item A003, Contract N62269-75-C-0503, NADC,1975 , section 5.0 for measured processing time results.

DATA RAM MEMORY MAP

ADDRESS	SIZE	TABLE	CONTENTS
000 - 087	88	TADIO	Input/Output storage area A
088 - 104	17	TRVOB	The number and positions of errata in the data characters (power form)
105 - 121	17	TRVOA	The total number and positions of erasures and errors (code form)
122 - 125	4		Unassigned
126 - 127	2	TADRW	Data Read/Write control flags, area A
128 - 215	88	TBDIO	Input/Output storage area B
216 - 232	17	TRVOC	The number of errata in the data characters and the product terms $\beta_i \prod (\beta_i + \beta_j)$ in power form
233 - 253	21		Unassigned
254 - 255	2	TBDRW	Data Read/Write control flags, area B
256 - 318	63	TPTCC	Code form of Galois field elements
319 - 319	1	SCBIX	Current data input/output block indicator
320 - 351	32	TCTPC	Power form of Galois field elements
352 - 368	17	TAPOL	Coefficients of erasure polynomial in power form
369 - 385	17	TMSYN	The number of erasures and the code form of the modified syndromes
386 - 402	17	TMSYP	The number of erasures and the power form of the modified syndromes
403 - 419	17		Unassigned
420 - 420	1	SEPEP	Address of the error polynomial
421 - 421	1		Unassigned
422 - 430	9	TEVAL	The number and positions of errors found
431 - 439	9	TEVOA	The coefficients of the error polynomial in power form

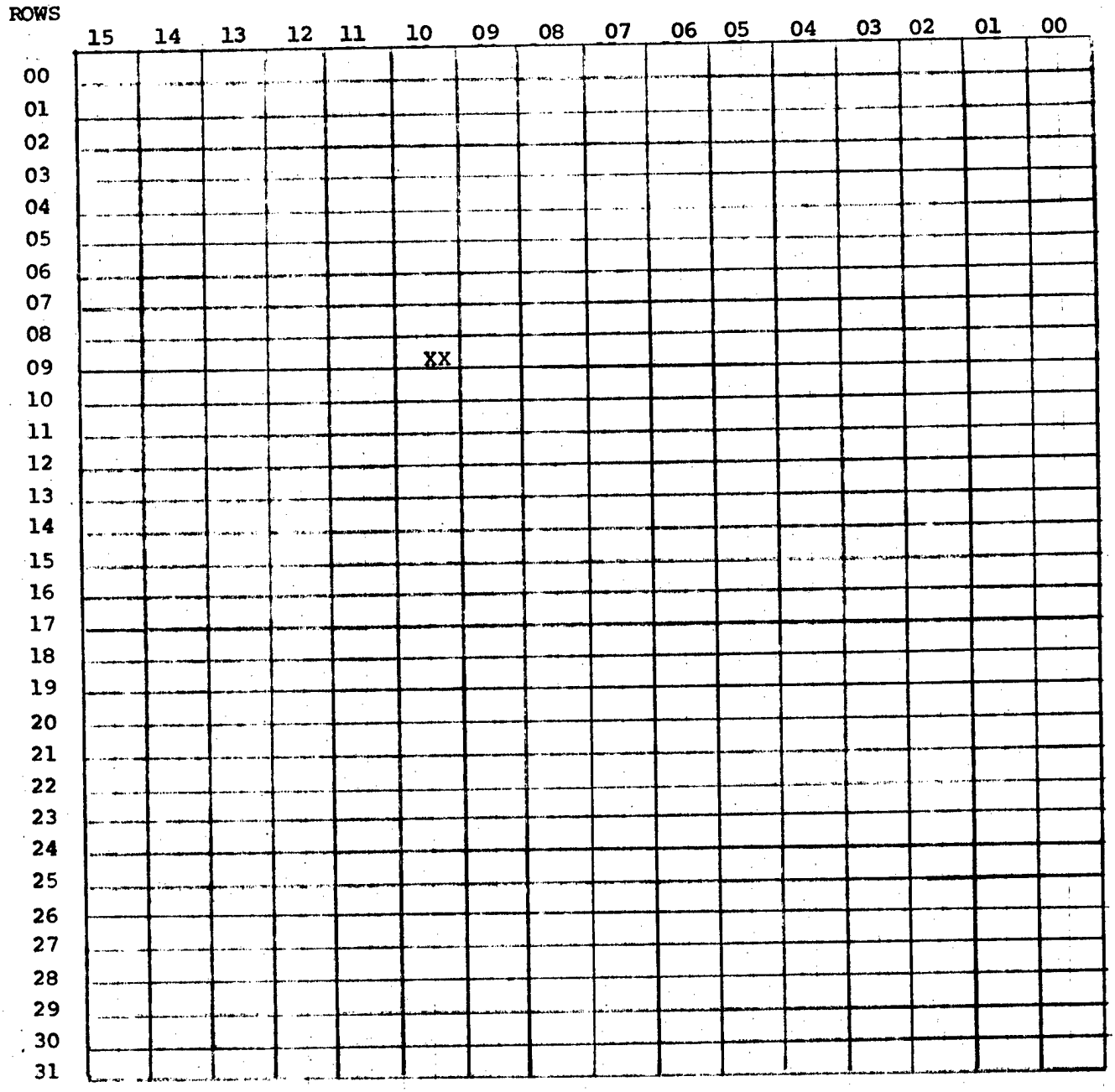
TABLE 3.2 Page 1/2

DATA RAM MEMORY MAP (Continued)

ADDRESS	SIZE	TABLE	CONTENTS
440 - 441	1	SIRCS	Galois sum (XOR) of inverse roots of the error polynomial in code form
441 - 441	1	SIRPS	Galois product (Sum) of inverse roots of the error polynomial in power form
442 - 447	6		Unassigned
448 - 456	9	TEPAA	Either the previous, current or next set of output terms computed by subprogram EPGEN
457 - 465	9	TEPBB	Similar to TEPAA
466 - 474	9	TEPCC	Similar to TEPAA
475 - 480	6	TEPDR	Used to store intermediate results by subprogram EPGEN
481 - 511	31		Unassigned

TABLE 3.2 Page 2/2

PROM ROW/COLUMN MEMORY MATRIX

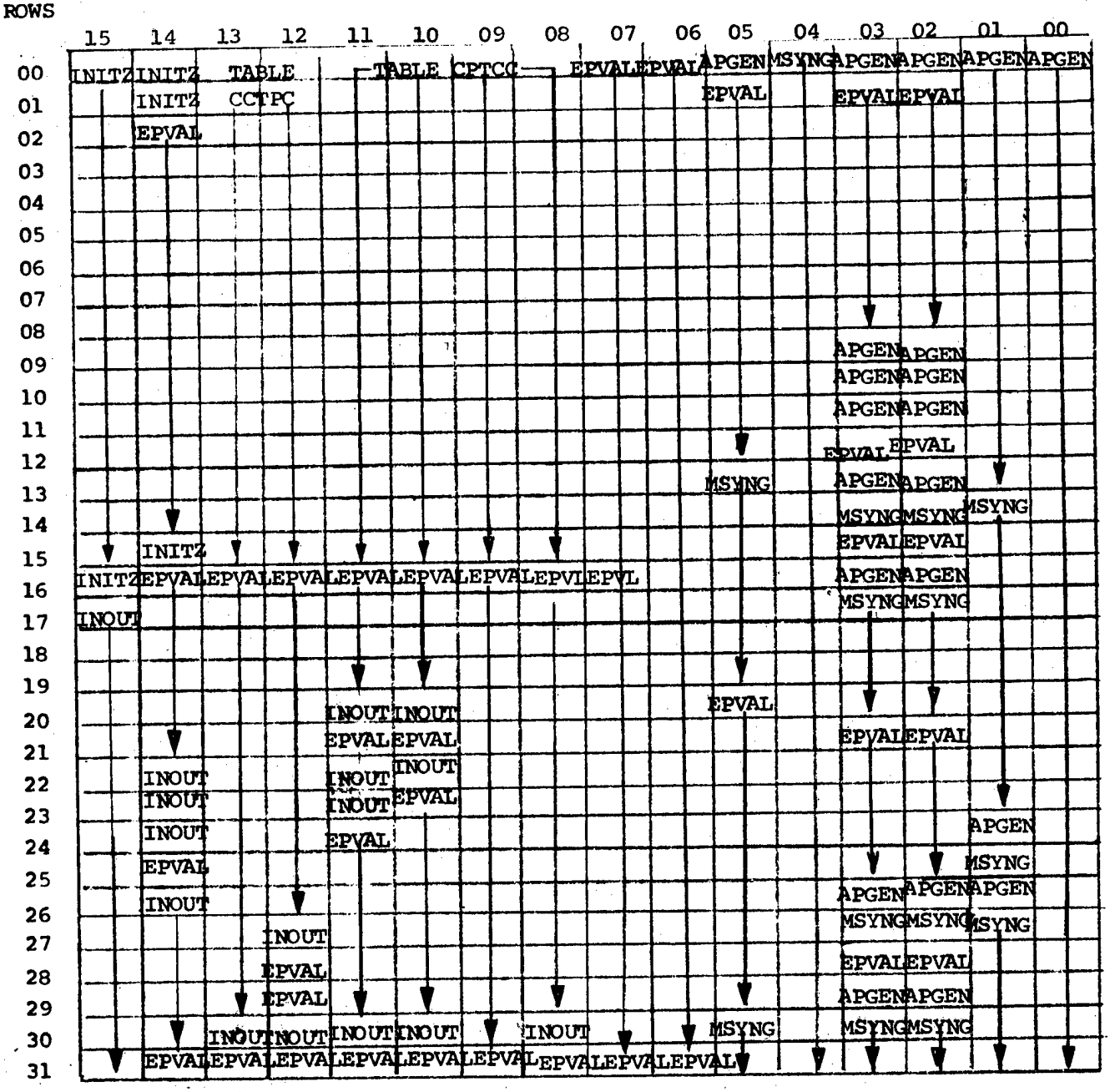


512 Instructions are stored in each ROM plane.

Address at XX is row 09, column 10

FIGURE 3.4

PROM MAP OF SUBPROGRAMS PLANE 00



COLUMNS

FIGURE 3. 5 Page 1/2

PROM MAP OF SUBPROGRAMS PLANE 01

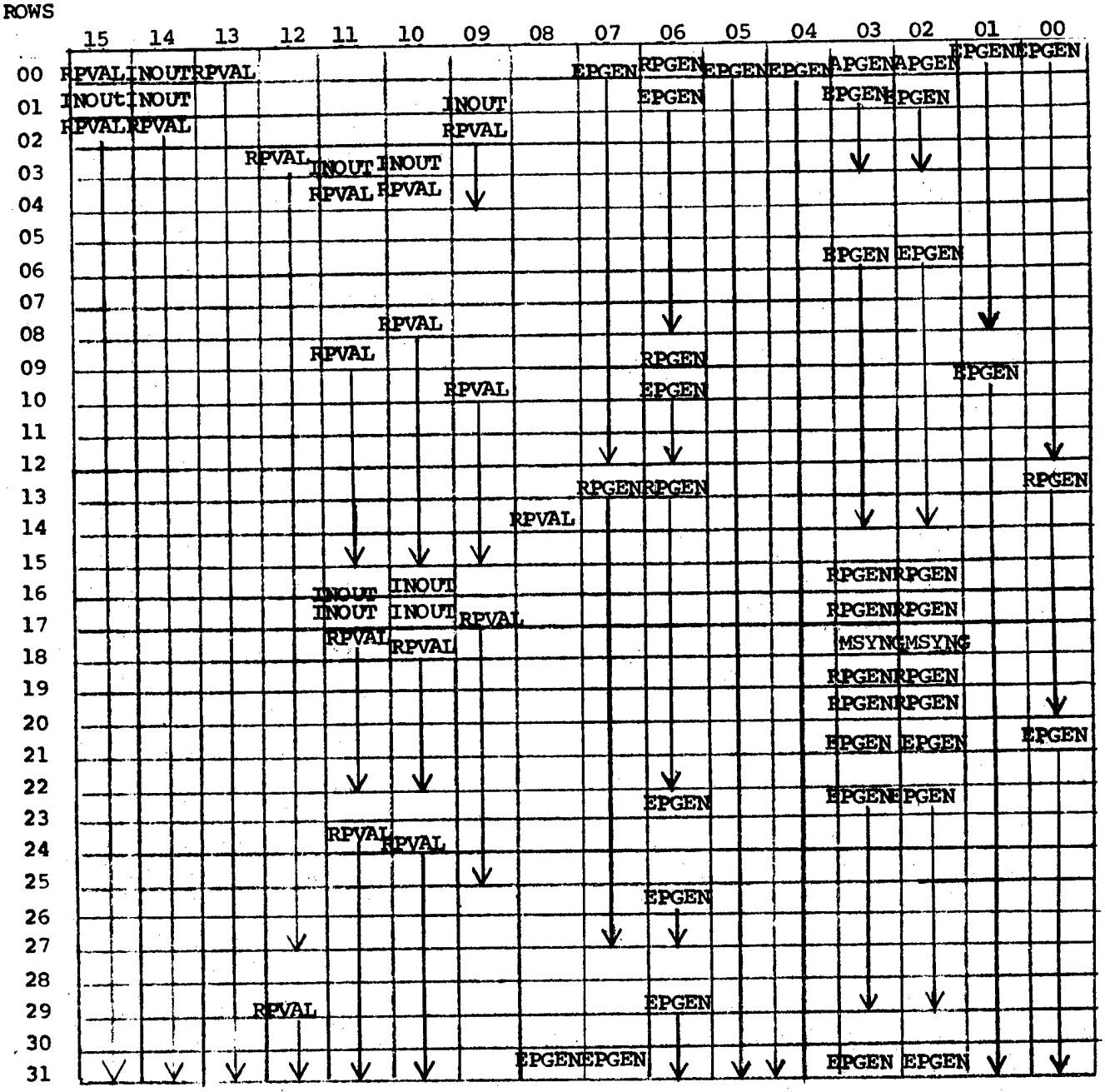


FIGURE 3.5 Page 2/2

3.4 Programming Guidelines

3.4.1 Programming language

The programming language to be used in RSED1 is the microinstruction set of micro-operations (MICROPS) of the INTEL 3000 microprocessor. The manuals to be used are the following:

- o 3001 Microprogram Control Unit — Reference 4, section 2.0
- o 3002 Central Processing Element, — Reference 5, section 2.0
- o Series 3000 Cross Microprogramming System, — Reference 6, section 2.0

Table 3.3 provides a summary of the 3000 micro-operations.

3.4.2 Program Loading

The initial loading of the subprograms of RSED1 will be a manual operation loading into a ROM Simulator. A TTY paper tape will be punched out from the ROM Simulator and will be used for future loadings. A new TTY paper tape will be punched out each time additions or changes are made to the program after loading it.

The final debugged version of RSED1 will be burned into the PROMs directly from the ROM Simulator.

3.4.3 Mnemonic Labeling Scheme

The following paragraphs describe the labeling scheme adopted in RSED1.

- o Program, Subprogram, Address Labels

  Five character labels will be used to designate the microprocessor program names, entrances, exits and other reference addresses. The labels selected are as follows:

PROGRAM	START ADDRESS	FUNCTION
RSED1		Total RSED program
INITZ	INI00	Initialization subprogram
INOUT	IOC00	Input/Output control
APGEN	APG00	Erasure Polynomial generator
MSYNG	MSY00	Modified Syndrome generator
EPGEN	EPG00	Error Polynomial generator
EPVAL	EPV00	Error Polynomial evaluator

MICRO-FUNCTION SUMMARY

F-GROUP	R-GROUP	MICRO-FUNCTION	
0	I	$R_n + (AC \wedge K) + CI \rightarrow R_n, AC$	
	II	$M + (AC \wedge K) + CI \rightarrow AT$	
	III	$AT_L \wedge (\overline{I_L \wedge K_L}) \rightarrow RO$	$LI \vee [(I_H \wedge K_H) \wedge AT_H] \rightarrow AT_H$ $[AT_L \wedge (I_L \wedge K_L)] \vee [AT_H \vee (I_H \wedge K_H)] \rightarrow AT_L$
1	I	$K \vee R_n \rightarrow MAR$	$R_n + K + CI \rightarrow R_n$
	II	$K \vee M \rightarrow MAR$	$M + K + CI \rightarrow AT$
	III	$(\overline{AT} \vee K) + (AT \wedge K) + CI \rightarrow AT$	
2	I	$(AC \wedge K) - 1 + CI \rightarrow R_n$	(see Note 1)
	II	$(AC \wedge K) - 1 + CI \rightarrow AT$	
	III	$(I \wedge K) - 1 + CI \rightarrow AT$	
3	I	$R_n + (AC \wedge K) + CI \rightarrow R_n$	
	II	$M + (AC \wedge K) + CI \rightarrow AT$	
	III	$AT + (I \wedge K) + CI \rightarrow AT$	
4	I	$CI \vee (R_n \wedge AC \wedge K) \rightarrow CO$	$R_n \wedge (AC \wedge K) \rightarrow R_n$
	II	$CI \vee (M \wedge AC \wedge K) \rightarrow CO$	$M \wedge (AC \wedge K) \rightarrow AT$
	III	$CI \vee (AT \wedge I \wedge K) \rightarrow CO$	$AT \wedge (I \wedge K) \rightarrow AT$
5	I	$CI \vee (R_n \wedge K) \rightarrow CO$	$K \wedge R_n \rightarrow R_n$
	II	$CI \vee (M \wedge K) \rightarrow CO$	$K \wedge M \rightarrow AT$
	III	$CI \vee (AT \wedge K) \rightarrow CO$	$K \wedge AT \rightarrow AT$
6	I	$CI \vee (AC \wedge K) \rightarrow CO$	$R_n \vee (AC \wedge K) \rightarrow R_n$
	II	$CI \vee (AC \wedge K) \rightarrow CO$	$M \vee (AC \wedge K) \rightarrow AT$
	III	$CI \vee (I \wedge K) \rightarrow CO$	$AT \vee (I \wedge K) \rightarrow AT$
7	I	$CI \vee (R_n \wedge AC \wedge K) \rightarrow CO$	$R_n \bar{\oplus} (AC \wedge K) \rightarrow R_n$
	II	$CI \vee (M \wedge AC \wedge K) \rightarrow CO$	$M \bar{\oplus} (AC \wedge K) \rightarrow AT$
	III	$CI \vee (AT \wedge I \wedge K) \rightarrow CO$	$AT \bar{\oplus} (I \wedge K) \rightarrow AT$

NOTES:

1. 2's complement arithmetic adds 111 . . . 11 to perform subtraction of 000 . . . 01.
2. R_n includes T and AC as source and destination registers in R-group 1 micro-functions.
3. Standard arithmetic carry output values are generated in F-group 0, 1, 2 and 3 instructions.

SYMBOL	MEANING
I, K, M	Data on the I, K, and M busses, respectively
CI, LI	Data on the carry input and left input, respectively
CO, RO	Data on the carry output and right output, respectively
R_n	Contents of register n including T and AC (R-Group I)
AC	Contents of the accumulator
AT	Contents of AC or T, as specified
MAR	Contents of the memory address register
L, H	As subscripts, designate low and high order bit, respectively
+	2's complement addition
−	2's complement subtraction
∧	Logical AND
∨	Logical OR
$\bar{\oplus}$	Exclusive-NOR
→	Deposit into

TABLE 3.3 Page 1/2

ALL-ZERO AND ALL-ONE K-BUS MICRO-FUNCTIONS

K-BUS = 00 MICRO-FUNCTION		MNEMONIC	K-BUS = 11 MICRO-FUNCTION		MNEMONIC
$R_n + CI \rightarrow R_n$, AC		ILR	$AC + R_n + CI \rightarrow R_n$, AC		ALR
$M + CI \rightarrow AT$		ACM	$M + AC + CI \rightarrow AT$		AMA
$AT_L \rightarrow RO$	$AT_H \rightarrow AT_L$ $LI \rightarrow AT_H$	SRA	(See Appendix B)		–
$R_n \rightarrow MAR$	$R_n + CI \rightarrow R_n$	LMI	$11 \rightarrow MAR$	$R_n - 1 + CI \rightarrow R_n$	DSM
$M \rightarrow MAR$	$M + CI \rightarrow AT$	LMM	$11 \rightarrow MAR$	$M - 1 + CI \rightarrow AT$	LDM
$\overline{AT} + CI \rightarrow AT$		CIA	$AT - 1 + CI \rightarrow AT$		DCA
$CI - 1 \rightarrow R_n$	See Note 1	CSR	$AC - 1 + CI \rightarrow R_n$	See Note 1	SDR
$CI - 1 \rightarrow AT$	See Note 1	CSA	$AC - 1 + CI \rightarrow AT$	See Note 1	SDA
(See CSA above)		–	$I - 1 + CI \rightarrow AT$		LDI
$R_n + CI \rightarrow R_n$		INR	$AC + R_n + CI \rightarrow R_n$		ADR
(See ACM above)		–	(See AMA above)		–
$AT + CI \rightarrow AT$		INA	$I + AT + CI \rightarrow AT$		AIA
$CI \rightarrow CO$	$0 \rightarrow R_n$	CLR	$CI \vee (R_n \wedge AC) \rightarrow CO$	$R_n \wedge AC \rightarrow R_n$	ANR
$CI \rightarrow CO$	$0 \rightarrow AT$	CLA	$CI \vee (M \wedge AC) \rightarrow CO$	$M \wedge AC \rightarrow AT$	ANM
(See CLA above)		–	$CI \vee (AT \wedge I) \rightarrow CO$	$AT \wedge I \rightarrow AT$	ANI
(See CLR above)		–	$CI \vee R_n \rightarrow CO$	$R_n \rightarrow R_n$	TZR
(See CLA above)		–	$CI \vee M \rightarrow CO$	$M \rightarrow AT$	LTM
(See CLA above)		–	$CI \vee AT \rightarrow CO$	$AT \rightarrow AT$	TZA
$CI \rightarrow CO$	$R_n \rightarrow R_n$	NOP	$CI \vee AC \rightarrow CO$	$R_n \vee AC \rightarrow R_n$	ORR
$CI \rightarrow CO$	$M \rightarrow AT$	LMF	$CI \vee AC \rightarrow CO$	$M \vee AC \rightarrow AT$	ORM
(See NOP above)		–	$CI \vee I \rightarrow CO$	$I \vee AT \rightarrow AT$	ORI
$CI \rightarrow CO$	$\overline{R_n} \rightarrow R_n$	CMR	$CI \vee (R_n \wedge AC) \rightarrow CO$	$R_n \oplus AC \rightarrow R_n$	XNR
$CI \rightarrow CO$	$\overline{M} \rightarrow AT$	LCM	$CI \vee (M \wedge AC) \rightarrow CO$	$M \oplus AC \rightarrow AT$	XNM
$CI \rightarrow CO$	$\overline{AT} \rightarrow AT$	CMA	$CI \vee (AT \wedge I) \rightarrow CO$	$I \oplus AT \rightarrow AT$	XNI

TABLE 3.3 Page 2/2

PROGRAM	START ADDRESS	FUNCTION
RPGEN	RPG00	Errata Polynomial generator
RPVAL	RPV00	Errata Polynomial evaluator

- Common Table Labels

Five character labels will be used to designate table names. Tables with variable content (stored in Data RAM) will be labeled TXXXX. Tables with fixed content (stored in PROM) will be labeled CXXXX. X is any alphanumeric.

- Internal Table Labels

Five character labels will be used to designate internal table names. The first character of each table will be the letter T. The second and third characters are chosen to identify the subprogram which uses the internal table. The characters selected are as follows:

SUBPROGRAM	TABLE NAMES
INITZ	TIZXX
INOUT	TIOXX
APGEN	TAPXX
MSYNG	TMSXX
EPGEN	TEPXX
EPVAL	TEVXX
RPGEN	TRPXX
RPVAL	TRVXX

- Item Labels

Five character labels will be used to designate item names. The first character of each item label will be the letter I, the second and third characters will be identical to the second and third characters of the name of the table to which the item belongs. The fourth and fifth characters may be given any alphanumeric value.

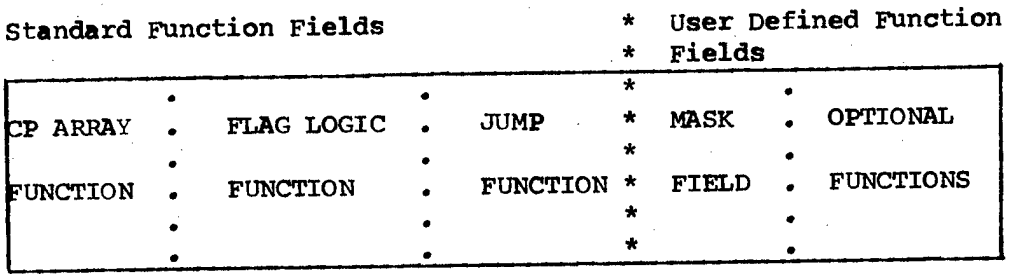
An item is defined as any subunit of a table (from 1 to N bits where N bits equals the total table size).

- Single Word Variables

  Five character labels,with the first character the letter S, will be used to identify single word variables.

- Microinstruction (Function Fields)

  A microinstruction consists of a number of microfunction fields as shown below:



The labels of the Standard Function Fields (Intrinsic Fields) are defined by INTEL as follows:

NAME	DESCRIPTION
CPE	CPE Array Function
FI	Flag Control Field- MCU Input
FO	Flag Control Field- MCU Output
JUMP	Jump Function (Next Address Control)

The labels of the User Defined Function Fields (Mask Field plus user options) are defined as follows:

NAME	DESCRIPTION
K	Mask Field (K bus)
II	Inhibit CPE Clock
LL	Load X bus as Address
PP	PROM Plane Select
RR	Data RAM Read/Write Access Control

o Micro-Operations (MICROPS)

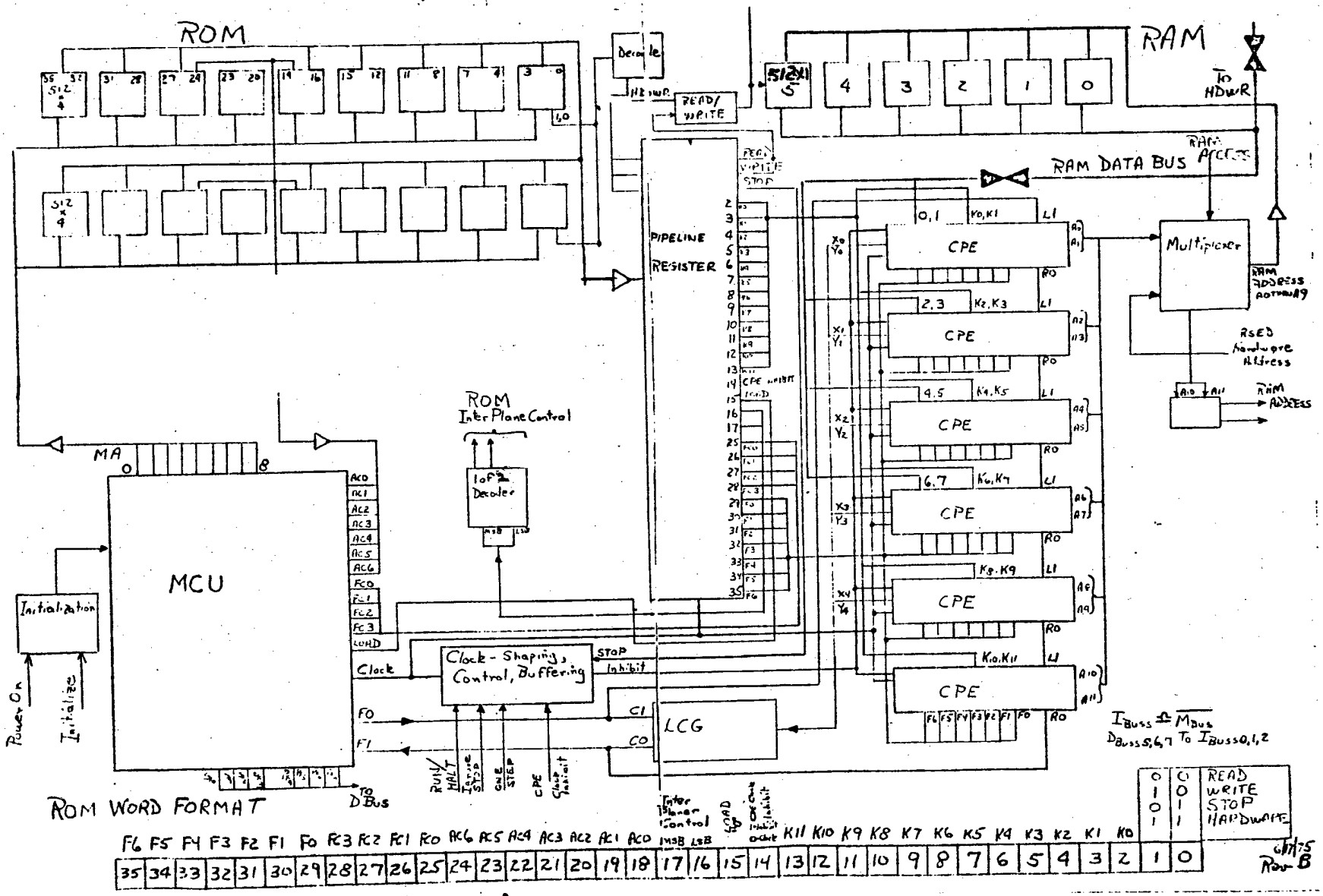
The contents of a function field are called MICROPS. Each field may have a set of MICROPS associated with it.

The sets of MICROPS for the Standard Function Fields (Intrinsic Fields) are defined by INTEL. The labels of these standard MICROPS together with a functional description are listed in Table 3.3 .

The sets of MICROPS for the User Defined Function Fields (Mask Field plus user options) are defined as follows:

NAME	DESCRIPTION
INH	Inhibit clock to CPE
LOAD	Load X bus as Address
PLAN0	Next Address is in PROM Plane 00
PLAN1	Next Address is in PROM Plane 01
READ	Read from Data RAM
WRITE	Write into Data RAM
STOP	Halt Microprocessor

Reference Figure 3.6 , RSED Microprocessor Block Diagram for the MICROPS word format.



RSED MICROPROCESSOR BLOCK DIAGRAM FIGURE 3.6

4.0 DETAILED REQUIREMENTS

4.1 Subprogram INITZ

4.1.1 Detailed Description

The overall function of subprogram INITZ is to perform all initialization functions required whenever the master (at power turn on) reset pulse or cyclic (at the start of each time slot) reset pulse is provided by the associated RSED hardware. The master and cyclic reset pulse forces the microprocessor to begin the execution of the instruction located at address row 0, column 15 . The first instruction of subprogram INITZ is located at this address.

The following functions are performed:

- o INITZ provides a lockout of the non-microprocessor hardware i.e. the Data Steering and Control Unit (DSCU) until all initialization functions are completed. It is necessary to prevent the DSCU from accessing the Data Ram until the data control flags have been properly set and a record of their current setting is stored. Refer to Figure 3.2 and Reference 1.

  The lockout is achieved by placing either a read or write option in each instruction. Priority to the Data Ram is given to the microprocessor so that a Data Ram request will always effectively keep the DSCU locked out from the Data Ram. Write instructions are used whenever INITZ requires to write into the Data Ram otherwise redundant Read instructions are used. The effect of a Read instruction is to load the M bus with data read from the Data Ram. The M bus is then ignored by the microprocessor when the Read is redundant.

- o INITZ sets the Data Read/Write control flags to request one block of data to be read into the Data Ram. The Write request flag of input/output area A is set to the value 01 and the Read request flag in area A is set to the value 0. In the input/output area B, the Write request flag is set to the value 00 and the Read request flag is set to the value 0.(No request for area B.) Reference Table 4.1

- o INITZ specifies which of the two data blocks will receive the first block of data by setting Register 9 to the value zero. Reference section 4.1.3 (Indexes). This setting is required for subprogram interfacing.

- o INITZ writes the code and power tables of Galois field elements into the Data Ram. These constants are stored in the microprocessor ROM but are transferred to the Data Ram to optimize their usage when converting between code and power.

  The coding which performs the table transfers makes use of the LOAD option which forces a jump into the table on a word by word basis. This enables the current word to be read from ROM via the K bus and in the same instruction a jump function back to the transfer loop is requested for the next instruction. See Reference 4.

MICROPROCESSOR READ/WRITE CONTROL

FLAG	MEANING	SET BY:
X = 0	Data not ready	DSCU (on read completion)
X = 1	Data ready	Microprocessor (on decode completion)
ZY = 00	No write request made	Microprocessor (when processing begins)
ZY = 01	Write request made (DSCU may write)	Microprocessor (when processing of a word is near completion)
ZY = 11	DSCU has filled input buffer	DSCU (on write completion)
ZY = 10	Not used	

NOTES

1. A set of Read/Write control flags exist for each input/output area in a microprocessor

2. The flags are stored in fixed locations in the Data Ram used by each microprocessor as follows:

RAM ADDRESS (decimal)				MNEMONIC
WORD	BIT			
126	0	X Flag	Area A	IADXX
127	1,0	zY Flag	Area A	IADZY
254	0	X Flag	Area B	IBDXX
255	1,0	ZY Flag	Area B	IBDZY

TABLE 4.1

4.1.2 Flow Diagrams

Figure 4.1 is the flow diagram for subprogram INITZ.

4.1.3 Environment

4.1.3.1 Tables

INITZ utilizes only tables in the common data base.

4.1.3.2 Variables

INITZ does not access any variables.

4.1.3.3 Constants

INITZ does not utilize any constants as such. Due to the structure of the 3000 MCU, Reference 4, the K bus can be used as a constant but it is inherently part of each instruction and has to be defined as such. Tables 4.2 and 4.3 define the contents of the Galois field element conversion tables which INITZ moves from ROM to RAM.

4.1.3.4 Flags

INITZ utilizes only common flags.

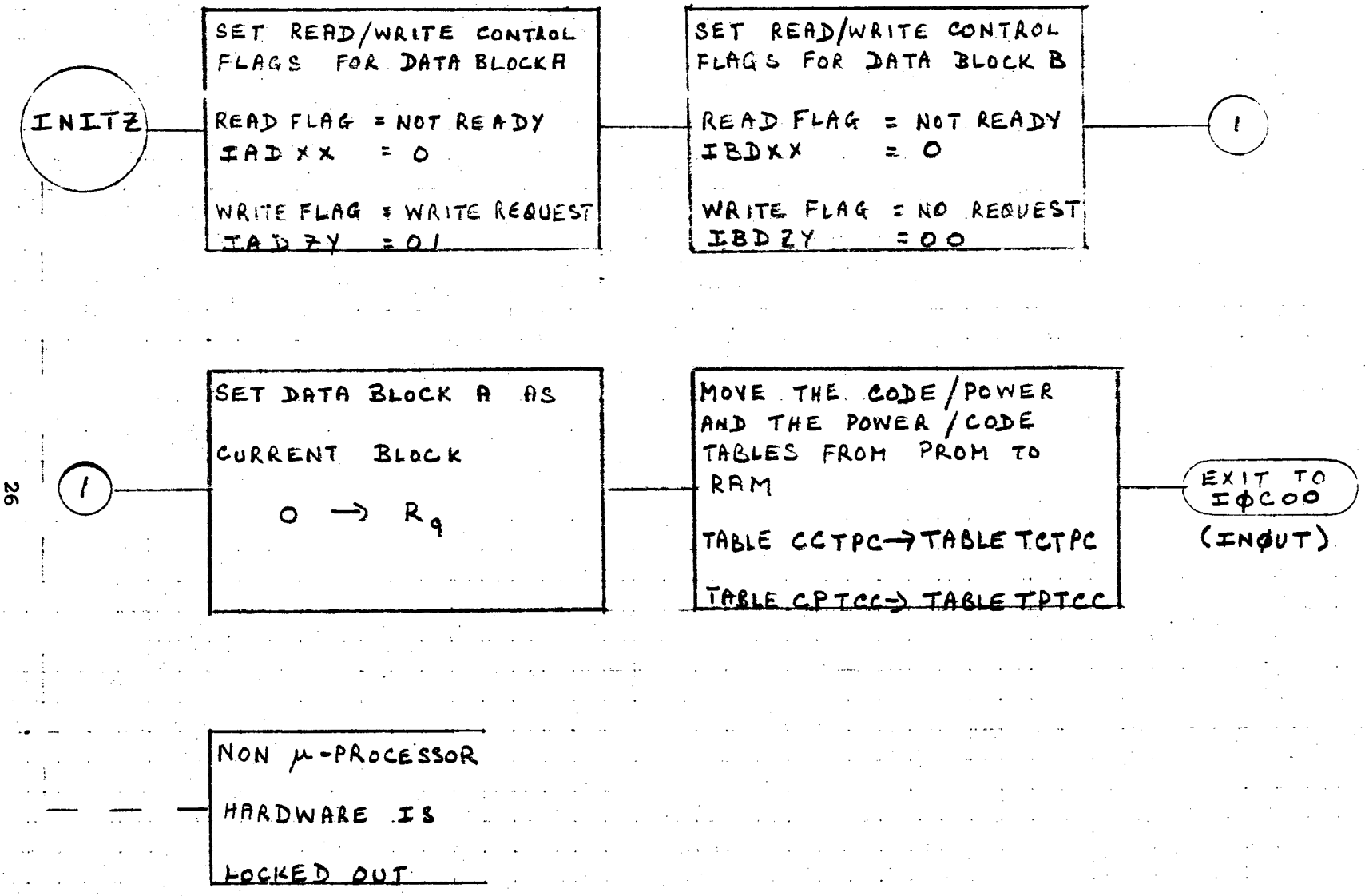
4.1.3.5 Indexes

Table 4.4 describes the indexes used by INITZ.

4.1.3.6 Common Data Base Reference

The following common data items are referenced by INITZ:

ITEM	TABLE (in Data Base)
IADXX	TADRW
IADZY	TADRW
IBDXX	TBDRW
IBDZY	TBDRW
TCTPC	TCTPC
TPTCC	TPTCC



SUBPROGRAM INITZ FLOW DIAGRAM

FIGURE 4.1

CODE TO POWER CONVERSION TABLE

CCTPC ROM ADDRESS (D)				TCTPC		RAM ADDRESS (O)	
R	C	P	BITS	BINARY	OCTAL	WORD	BITS
00	12	00	6-2	00000	00	500	4-0
01	12	00	6-2	11111	37	501	4-0
02	12	00	6-2	00001	01	502	4-0
03	12	00	6-2	10010	22	503	4-0
04	12	00	6-2	00010	02	504	4-0
05	12	00	6-2	00101	05	505	4-0
06	12	00	6-2	10011	23	506	4-0
07	12	00	6-2	01011	13	507	4-0
08	12	00	6-2	00011	03	510	4-0
09	12	00	6-2	11101	35	511	4-0
10	12	00	6-2	00110	06	512	4-0
11	12	00	6-2	11011	33	513	4-0
12	12	00	6-2	10100	24	514	4-0
13	12	00	6-2	01000	10	515	4-0
14	12	00	6-2	01100	14	516	4-0
15	12	00	6-2	10111	27	517	4-0
00	13	00	6-2	00100	04	520	4-0
01	13	00	6-2	01010	12	521	4-0
02	13	00	6-2	11110	36	522	4-0
03	13	00	6-2	10001	21	523	4-0
04	13	00	6-2	00111	07	524	4-0
05	13	00	6-2	10110	26	525	4-0
06	13	00	6-2	11100	34	526	4-0
07	13	00	6-2	11010	32	527	4-0
08	13	00	6-2	10101	25	530	4-0
09	13	00	6-2	11001	31	531	4-0
10	13	00	6-2	01001	11	532	4-0
11	13	00	6-2	10000	20	533	4-0
12	13	00	6-2	01101	15	534	4-0
13	13	00	6-2	01110	16	535	4-0
14	13	00	6-2	11000	30	536	4-0
15	13	00	6-2	01111	17	537	4-0

TABLE 4.2

POWER TO CODE CONVERSION TABLE

CPTCC ROM ADDRESS (D)				CONTENT		TPTCC RAM ADDRESS (O)	
R	C	P	BITS	BINARY	OCTAL	WORD	BITS
00	08	00	6-2	00000	00	400	4-0
01	08	00	6-2	00010	02	401	4-0
02	08	00	6-2	00100	04	402	4-0
03	08	00	6-2	01000	10	403	4-0
04	08	00	6-2	10000	20	404	4-0
05	08	00	6-2	00101	05	405	4-0
06	08	00	6-2	01010	12	406	4-0
07	08	00	6-2	10100	24	407	4-0
08	08	00	6-2	01101	15	410	4-0
09	08	00	6-2	11010	32	411	4-0
10	08	00	6-2	10001	21	412	4-0
11	08	00	6-2	00111	07	413	4-0
12	08	00	6-2	01110	16	414	4-0
13	08	00	6-2	11100	34	415	4-0
14	08	00	6-2	11101	35	416	4-0
15	08	00	6-2	11111	37	417	4-0
00	08	00	6-2	11011	33	420	4-0
01	09	00	6-2	10011	23	421	4-0
02	09	00	6-2	00011	03	422	4-0
03	09	00	6-2	00110	06	423	4-0
04	09	00	6-2	01100	14	424	4-0
05	09	00	6-2	11000	30	425	4-0
06	09	00	6-2	10101	25	426	4-0
07	09	00	6-2	01111	17	427	4-0
08	09	00	6-2	11110	36	430	4-0
09	09	00	6-2	11001	31	431	4-0
10	09	00	6-2	10111	27	432	4-0
11	09	00	6-2	01011	13	433	4-0
12	09	00	6-2	10110	26	434	4-0
13	09	00	6-2	01001	11	435	4-0
14	09	00	6-2	10010	22	436	4-0
15	09	00	6-2	00001	01	437	4-0

TABLE 4.3 Page 1/2

POWER TO CODE CONVERSION TABLE

CPTCC ROM ADDRESS (D)				CONTENT		TPTCC RAM ADDRESS (O)	
R	C	P	BITS	BINARY	OCTAL	WORD	BITS
00	10	00	6-2	00010	02	440	4-0
01	10	00	6-2	00100	04	441	4-0
02	10	00	6-2	01000	10	442	4-0
03	10	00	6-2	10000	20	443	4-0
04	10	00	6-2	00101	05	444	4-0
05	10	00	6-2	01010	12	445	4-0
06	10	00	6-2	10100	24	446	4-0
07	10	00	6-2	01101	15	447	4-0
08	10	00	6-2	11010	32	450	4-0
09	10	00	6-2	10001	21	451	4-0
10	10	00	6-2	00111	07	452	4-0
11	10	00	6-2	01110	16	453	4-0
12	10	00	6-2	11100	34	454	4-0
13	10	00	6-2	11101	35	455	4-0
14	10	00	6-2	11111	37	456	4-0
15	10	00	6-2	11011	33	457	4-0
00	11	00	6-2	10011	23	460	4-0
01	11	00	6-2	00011	03	461	4-0
02	11	00	6-2	00110	06	462	4-0
03	11	00	6-2	01100	14	463	4-0
04	11	00	6-2	11000	30	464	4-0
05	11	00	6-2	10101	25	465	4-0
06	11	00	6-2	01111	17	466	4-0
07	11	00	6-2	11110	36	467	4-0
08	11	00	6-2	11001	31	470	4-0
09	11	00	6-2	10111	27	471	4-0
10	11	00	6-2	01011	13	472	4-0
11	11	00	6-2	10110	26	473	4-0
12	11	00	6-2	01001	11	474	4-0
13	11	00	6-2	10010	22	475	4-0
14	11	00	6-2	00001	01	476	4-0
15	11	00	6-2	00010	02	Not transferred.	

TABLE 4.3 Page 2/2

CPE REGISTERS USED AS INDEXES BY : INITZ

REGISTER	CONTENT/PURPOSE
R 0	
R 1	ROM address of current item in Table CCTPC/CPTCC
R 2	RAM address of current item in Table TCTPC/TPTCC
R 3	
R 4	
R 5	
R 6	
R 7	
R 8	
R 9	Current Data Block Indicator. Equals 0 when A is current Data Block Equals 200(Ø)when B is current Data Block
T	

TABLE 4.4

4.1.4 Input/Output Formats

Inputs and outputs processed by INITZ are listed below with a reference containing their description:

ITEM	INPUT	OUTPUT	REFERENCE
IADXX		X	Table 4.1
IADZY		X	Table 4.1
IBDXX		X	Table 4.1
IBDZY		X	Table 4.1
R_9		X	Table 4.4
TCTPC	X	X	Table 4.2
TPTPC	X	X	Table 4.3

4.1.5 Conditions for Initiation.

INITZ is initiated under hardware control whenever a master or cyclic reset pulse is given. A master reset pulse occurs at power turn on. A cyclic reset pulse occurs at the start of each time slot which is a synchronized event controlled by the external system.

4.1.6 Limitations.

INITZ does not have any special limitations.

4.1.7 Interface Description

After performing its functions INITZ exits to subprogram INOUT at entry point IOC00. Reference Figures 3.3 and 4.1 .

4.2 Subprogram INOUT

4.2.1 Detailed Description

Subprogram INOUT is the control program which communicates with the Data Steering and Control Unit (DSCU) to see when the processing cycle for one R/S code word may begin. There are three types of entrances from other subprograms into INOUT. Reference section 4.2.7 . The functions performed by INOUT at each entry point are descibed as follows:

- o Entry by subprogram INITZ at IOC00 for delay loop.

  INOUT provides a delay loop which is entered whenever the microprocessor is waiting to receive a new R/S code word. The first step in the delay processing is performed once only. This function is an initializing function which is required before a new R/S code word can be processed. Two single word variables SIRCS and SIRPS used by subprogram EPVAL are set to the value zero. Reference section 5.2 for a definition of the variables. Tables TAPAA, TAPBB and TAPCC are cleared to zero. Word 0 of Table TEPDR is set to 37(0) and word 4 of Table TEPDR is set to 1. Subprogram EPGEN requires these table settings. Reference section 5.1 for a description of these tables.

  INOUT then reads the current WRITE control flag from the Data Ram and checks to see if a new R/S code word has been written into the Data Ram by the interfacing DSCU. If a code word is not ready, INOUT sets the delay counter to the value zero. It then checks to see if the counter has reached its limit of 4. This limit corresponds to a delay cycle of 5 microseconds. If the limit has not been reached, the delay counter is incremented by one and the limit check is repeated. This processing continues until the delay limit is reached. When this occurs, INOUT again reads the WRITE flag from the Data Ram and rechecks it. If a new code word has not been received , the whole process is repeated. When a new code word has been received, INOUT sets the WRITE control flag to indicate that No Write Request is Made. The value of 20(0) is then written into the Data Quality word (word 07) of the current input/output area. INOUT then exits to subprogram APGEN at APG00 so that processing of the new code word can begin.

- o Entry by subprogram RPVAL after a sucessful decode occurs.

  After decode completion, INOUT sets the READ control flag for the current input/output area to indicate Data Ready to the interfacing DSCU. The current input/output area is then switched to the previous noncurrent area. INOUT then enters the delay cycle described above to await the arrival of the next R/S code word. This entry point is IOC03.

- Entry by other subprograms for a decode failure or an R/S code word without errata.

  At this entry point,INOUT restores the current block indicator which may by now have been destroyed by the entering subprogram. This can occur since the indicator is stored in the CPE register R9 which may be overwritten as a time saving device. The restoration is done by reading the current block indicator SCBIX, stored in the Data Ram. Reference section 5.2 .

  When subprograms enter INOUT at this point (IOC01) the normal processing path of a sucessful decode has not been followed and therefore the setting of the WRITE request control flag will not have been performed. Subprogram RPVAL sets the WRITE request flag for a next R/S code word before the processing of the first R/S code word is completed. This permits the transfer of some or all of the next R/S code word into the Data Ram before the microprocessor is ready to begin processing the next word. INOUT then exits to subprogram RPVAL at entry point RPV02. This is the point in RPVAL where it sets the WRITE request flag. A return is then made at once to INOUT at the same point, IOC03 at which subprogram RPVAL enters after a sucessful decode. The READ control flag setting, the input/output current indicator switching functions are performed and then the delay loop is entered. The exit from INOUT to RPVAL and the return to INOUT is done to optimize the processing time of the normal but longer path of a successful decode.

4.2.2 Flow Diagram

Figure 4.2 is the flow diagram for subprogram INOUT.

4.2.3 Environment

4.2.3.1 Tables

INOUT utilizes only tables in the common data base.

4.2.3.2 Variables

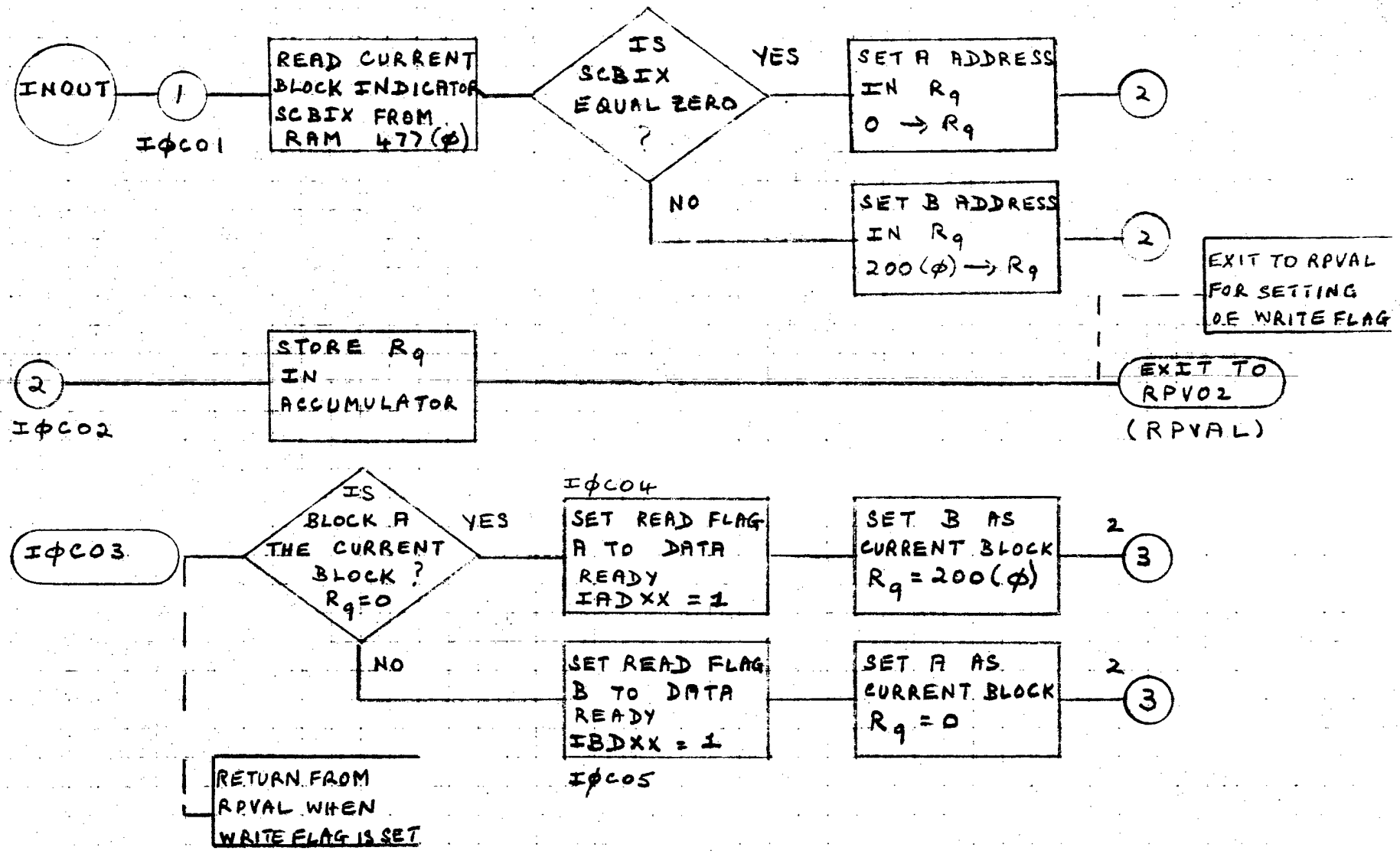
INOUT accesses only common data base variables.

4.2.3.3 Constants

INOUT does not use any constants other than the use of the K bus which is defined in the appropriate instruction.

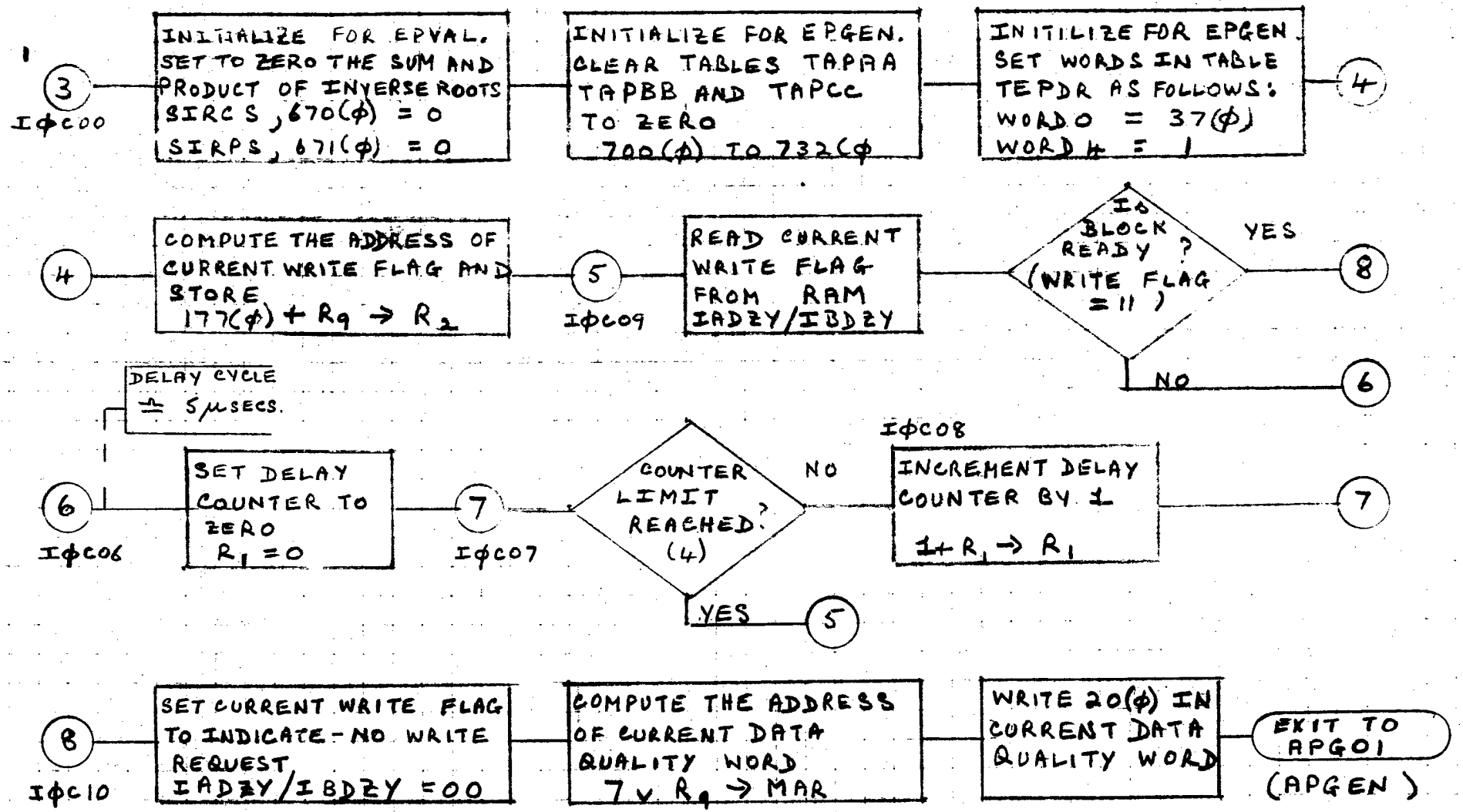
4.2.3.4 Flags

INOUT uses only common data base flags.



SUBPROGRAM INOUT FLOW DIAGRAM

FIGURE 4.2 Page 1/2



SUBPROGRAM INOUT FLOW DIAGRAM

FIGURE 4.2 Page 2/2

4.2.4 Input/Output Formats

Inputs and outputs processed by INOUT are listed below with a common data base reference for the description. Refer to Figure 4.2 for a description of item settings performed by INOUT.

ITEM	INPUT	OUTPUT	REFERENCE
IAD07		X	TADIO
IBD07		X	TBDIO
IADXX		X	TADRW
IADZY	X	X	TADRW
IBDXX		X	TBDRW
IBDZY	X	X	TBDRW
SCBIX	X		SCBIX
SIRCS		X	SIRCS
SIRPS		X	SIRPS
TEPAA Words 0 to 8		X	TEPOL
TEPBB Words 0 to 8		X	TEPOL
TEPCC Words 0 to 8		X	TEPOL
TEPDR Words 0 and 4		X	TEPDR
R_9	X	X	Table 4.5

4.2.5 Conditions for Initiation

INOUT is the control program and is always initiated. Refer to sections 4.2.1 and 4.2.7 for a complete description of all entry details.

4.2.6 Limitations

INOUT does not have any special limitations.

4.2.3.5 Indexes

Table 4.5 describes the indexes used by INOUT.

4.2.3.6 Common Data Base Reference

The following common data base items are referenced by INOUT:

ITEM	TABLE (in Data Base)
IAD07	TADIO
IBD07	TBDIO
IADXX	TADRW
IADZY	TADRW
IBDXX	TBDRW
IBDZY	TBDRW
SCBIX	SCBIX

Words 0 to 8	TEPAA	TEPOL
Words 0 to 8	TEPBB	
Words 0 to 8	TEPCC	

CPE REGISTERS USED AS INDEXES BY : **INOUT**

REGISTER	CONTENT/PURPOSE
R 0	
R 1	**Delay counter**
R 2	**RAM address of current Data Block Write control flag**
R 3	
R 4	
R 5	
R 6	
R 7	
R 8	
R 9	**Current Data Block Indicator.** **Equals 0 when A is current Data Block** **Equals 200(ø) when B is current Data Block**
T	

TABLE 4.5

4.2.7 Interface Description

Reference Figure 3.3 for a block diagram showing the sequential and functional relationship of all other subprograms with INOUT.

A description of all interfacing entrances and exits into and out of INOUT are presented below:

ENTERING SUBPROGRAM	ENTRY LABEL	EXIT	CONDITION
INITZ	IOC00		Initialization by hardware.
APGEN	IOC01		Decode Failure, $E > 16$.
MSYNG	IOC01		No errata in R/S code word. $e = E = 0$
EPGEN	IOC01		Decode Failure, $2e + E > 16$.
EPVAL	IOC01		Decode Failure, insufficient roots.
RPVAL	IOC03		Decode completion .
APGEN		X	WRITE control flag indicates a new R/S code word is ready for processing .
RPVAL		XX	When WRITE control flag has not been set to request new word. Occurs when IOC01 entry is made.
RPVAL	IOC03		After the exit XX occurs. (See above).

4.3 Subprogram APGEN

4.3.1 Detailed Description

Inputs to subprogram APGEN are the erasure locations in power form from the Syndrome Generator. Reference Figure 3.1 . APGEN computes the erasure polynomial which is defined as the following product:

$$\prod_{i=1}^{s} (1 + \beta_{E_i} X) \qquad \text{for all } i = 1, \ldots\ldots\ldots\ldots s$$

where s = number of erasures

and E_i = erasure locations in power form

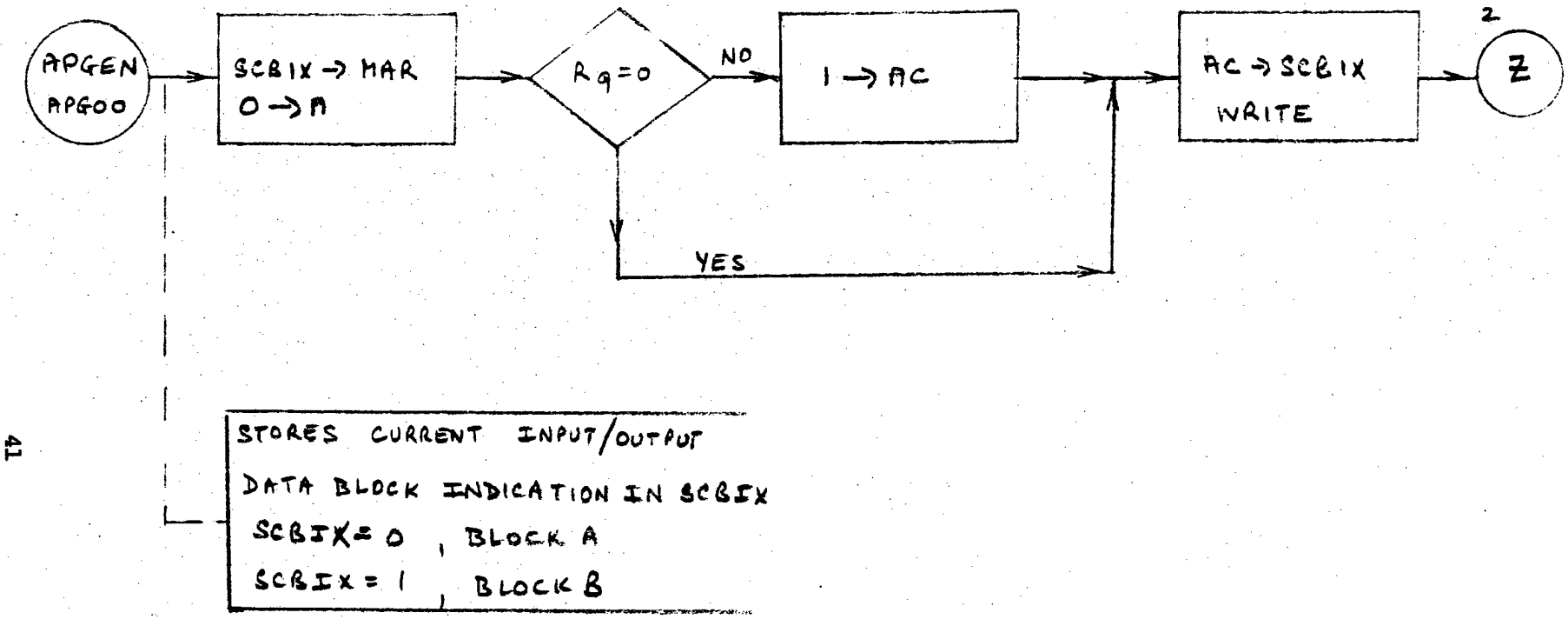
APGEN sucessively multiplies the previous partial product $(1 + \beta_i X)$ by each new β_i as i runs from 1 to s . The first value of the polynomial is therefore $(1 + \beta_1 X)$. Each succeeding version of the polynomial is effectively the sum of the current version and β times the current version. Therefore each term of a particular partial polynomial is used in the computation of two terms in the partial polynomial which follows it. Each coefficient is added to the coefficient of the same degree in the next polynomial as well as being multiplied by the new β and added to the coefficient one degree higher in the next partial polynomial.

APGEN writes each partial polynomial in the Data Ram in Table TAPOL. When APGEN has computed the last partial polynomial i.e. when all values of i have been taken it exits to subprogram MSYNG.

APGEN performs two house-keeping functions before it begins the major computation. APGEN first writes the current data block indicator, stored in R9, into the Data Ram, item SCBIX. APGEN also checks the number of erasures, an input item from the Syndrome Generator. If the number of erasures is less than seventeen (17),the computation described above is performed. If more than sixteen (16) erasures are present, then a decode failure has occurred and APGEN makes an abnormal exit to subprogram INOUT at entry point IOC01. Refer to section 4.2 .

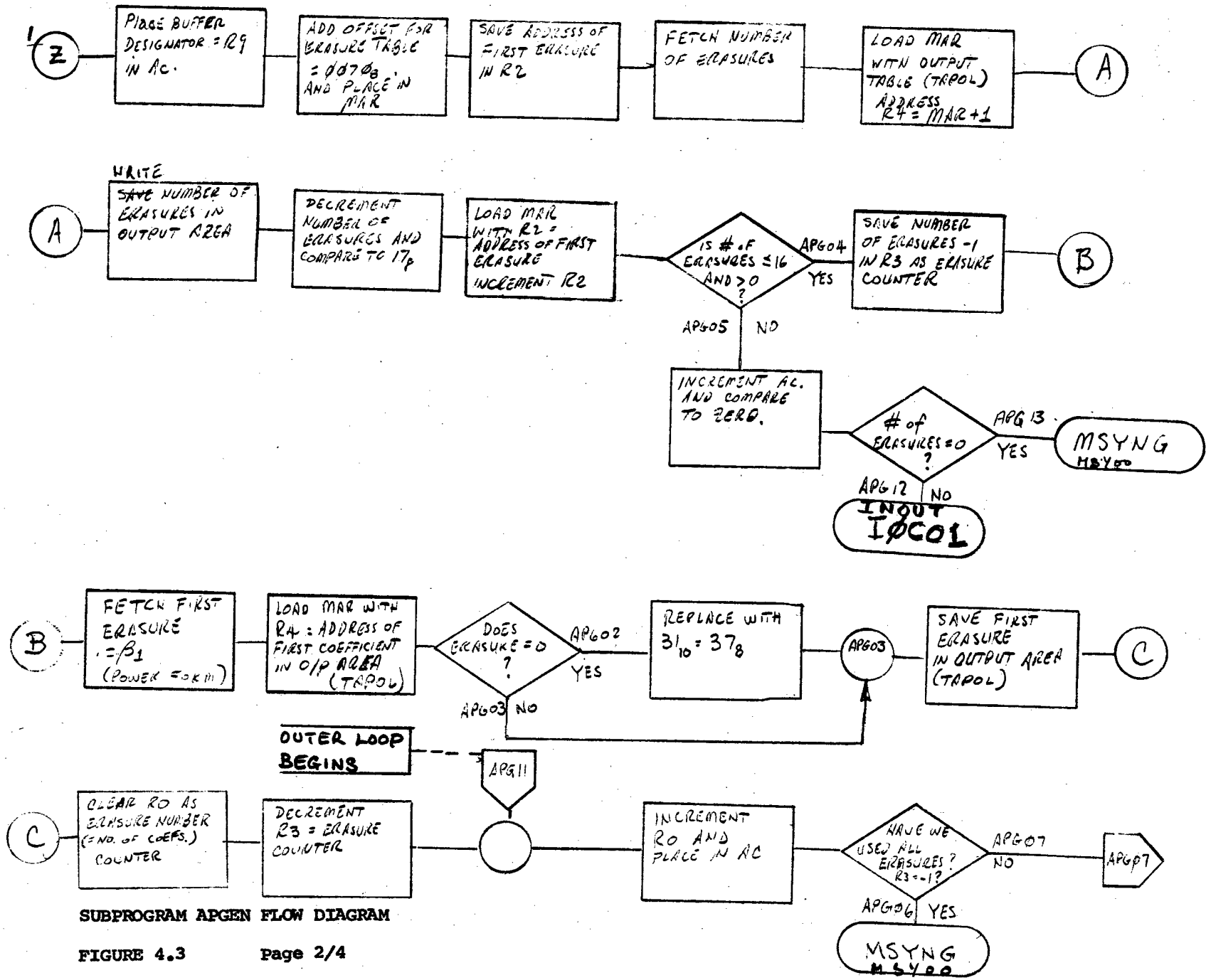
4.3.2 Flow Diagram

Figure 4.3 is the flow diagram for subprogram APGEN



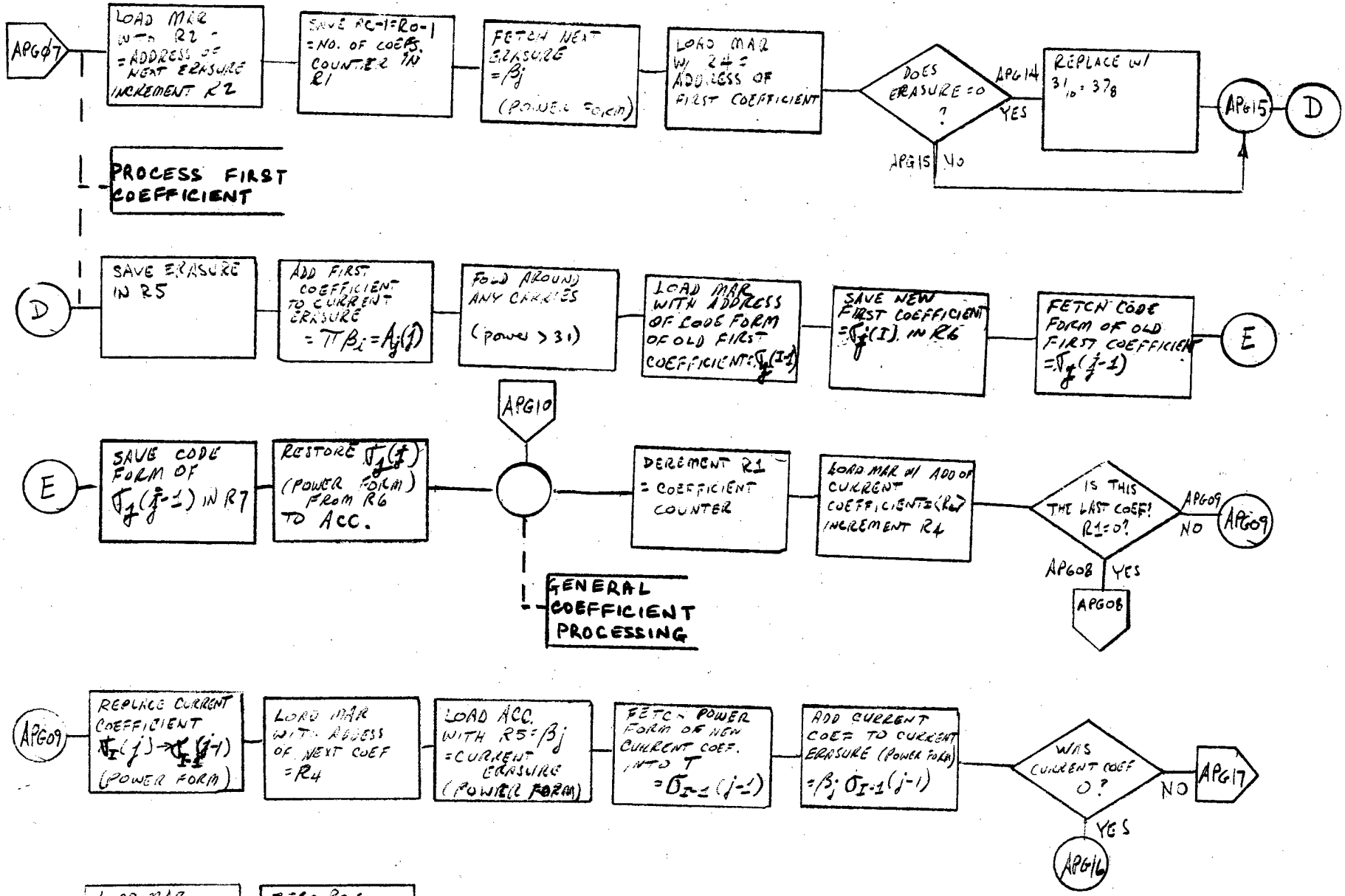
SUBPROGRAM APGEN FLOW DIAGRAM

FIGURE 4.3 Page 1/4



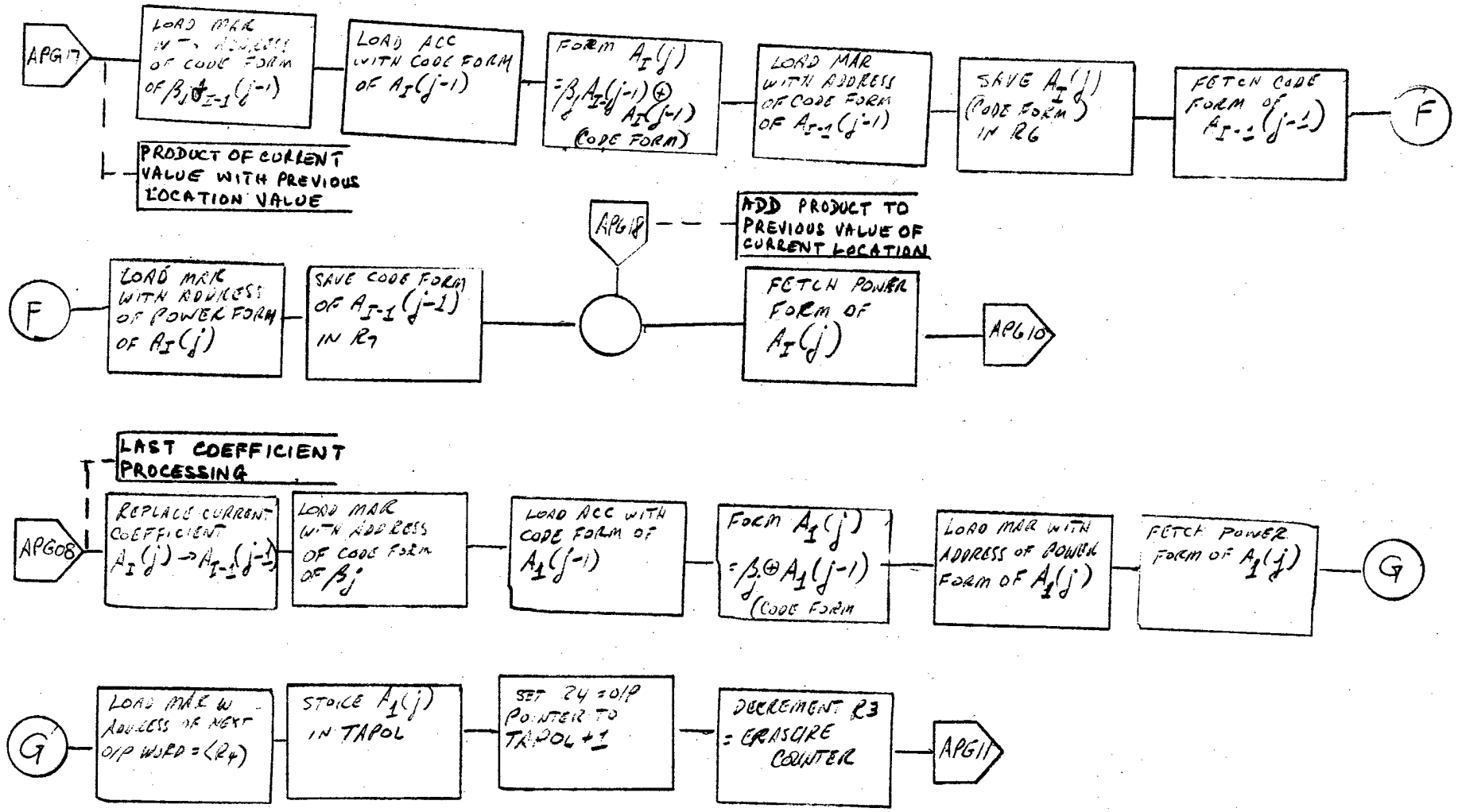
SUBPROGRAM APGEN FLOW DIAGRAM

FIGURE 4.3 **Page 2/4**



SUBPROGRAM APGEN FLOW DIAGRAM

FIGURE 4.3 **Page 3/4**



SUBPROGRAM APGEN FLOW DIAGRAM

FIGURE 4.3 Page 4/4

4.3.3 Environment

4.3.3.1 Tables

APGEN utilizes only common data base tables.

4.3.3.2 Variables

APGEN accesses only common data base variables.

4.3.3.3 Constants

APGEN does not use any constants other than the use of the K bus which is defined in the appropriate instruction.

4.3.3.4 Flags

APGEN uses only common data base flags.

4.3.3.5 Indexes

Table 4.6 describes the indexes used by APGEN

4.3.3.6 Common Data Base Reference

The following data base items are referenced by APGEN:

ITEM	TABLE
IADEE + 0	TADIO
•	
•	
IADEE + 30	
IBDEE + 0	TBDIO
•	
•	
IBDEE + 30	
IADES	TADIO
IBDES	TBDIO
SCBIX	SCBIX
TCTPC	TCTPC
TPTCC	TPTCC
TAPOL + 0	TAPOL
•	
•	
+ 16	

CPE REGISTERS USED AS INDEXES BY : APGEN

REGISTER	CONTENT/PURPOSE
R 0	Stores the value to which R_1 must be set at the start of processing next erasure. R_0 is incremented by 1.
R 1	The counter for the number of passes through the loop to compute all coefficients except the first and last.
R 2	Address pointer to erasures in current data block.
R 3	Counter for erasures.
R 4	Address pointer to output table TAPOL.
R 5	Current erasure in power form.
R 6	Current coefficient in code form.
R 7	Code form of the previous value in the previous output location.
R 8	
R 9	Current input/output data block. Equals 0 when A is current Data Block Equals 200(Ø)when B is current Data Block
T	Used to test coefficients for zero value.

TABLE 4.6

4.3.4 Input/Output Formats

Inputs and outputs processed by APGEN are listed below together with their common data base references which describe them in detail:

ITEM	INPUT	OUTPUT	REFERENCE
IADEE + 0 to IADEE + 30	X		TADIO
IBDEE + 0 to IBDEE + 30	X		TBDIO
IADES	X		TADIO
IBDES	X		TBDIO
SCBIX		X	SCBIX
TCTPC	X		TCTPC
TPTCC	X		TPTCC
TAPOL + 0 to TAPOL + 16		X	TAPOL
R_9	X		Table 4.6

4.3.5 Conditions for Initiation

Whenever subprogram INOUT finds a new R/S code word has been written into the Data Ram it passes control to APGEN.

4.3.6 Limitations

APGEN does not have any special limitations.

4.3.7 Interface Description

Reference Figure 3.3 for a block diagram showing the sequential and functional relationship of all subprograms.

APGEN computes the coefficients of the erasure polynomial which it stores in table TAPOL and then passes control to subprogram MSYNG. If there are no erasures, APGEN stores the value of zero in the first word of table TAPOL and then exits to subprogram MSYNG at the same entry point of MSY00 as before. If there are too many erasures (E>16) APGEN makes an abnormal exit to INOUT at IOC01 for a decode failure. This returns control to subprogram INOUT.

4.4 Subprogram MSYNG

4.4.1 Detailed Description

The major function of subprogram MSYNG is the computation of the modified syndromes which are defined as the set of terms T_j derived from the product of the syndromes and the coefficients of the erasure polynomial. The terms T_j are defined as follows:

$$T_j = \sum_{i=0}^{s} S_{s+j-i} \sigma_{E_i}$$

where S_i = syndromes, $i = 1,2,\ldots\ldots\ldots\ldots 16$

and σ_{E_i} = erasure polynomial coefficients, $i = 0,1,\ldots\ldots\ldots\ldots s$, s = number of erasures

and $j = 1,2,\ldots\ldots\ldots (16-s)$

The two inputs which MSYNG requires to perform this computation are the syndromes, generated by the RSED hardware Syndrome Generator and stored in the current input/output data block and the erasure polynomial coefficients, generated by subprogram APGEN and stored in table TAPOL.

MSYNG first moves the syndromes , input in code form and stores them, in code form in table TMSYN. At the same time MSYNG converts the syndromes from code to power form and stores the power form in the input/output data block i.e. table TADIO/TBDIO.

MSYNG then checks to see if all the syndromes equal zero. If this is the case, then by definition the R/S code word has no errata and needs no decoding. In this case , MSYNG sets the Data Quality word in the current input/output data block to the value zero. This indicates that the R/S code word was without error. MSYNG then exits to subprogram INOUT at entry point IOC01. Reference section 4.2.1 for subsequent processing details.

If the syndromes are not all equal to zero, MSYNG proceeds to generate the modified syndromes. Table TMSYN, the output table, already contains the syndromes in code form . The first syndrome is in word one etc. and the sixteenth syndrome is in word sixteen. Reference Table 4.7.

In the next pass the first coefficient σ_1 is added to S_1 in word 1. S_1 is then multiplied by σ_1 and added to S_2 in word 2.

S_2 is multiplied by σ_1 and added to S_3 in word 3. This continues

MODIFIED SYNDROME GENERATION

TAPOL	PASS 1	PASS 2	PASS 3	..... PASS s
Word 0				
Word 1	S_1	$S_1 + \sigma_1$	$S_1+\sigma_1$	
Word 2	S_2	$S_2 + S_1\sigma_1$	$S_2 +S_1\sigma_1+\sigma_2$	
Word 3	S_3	$S_3 + S_2\sigma_1$	$S_3 +S_2\sigma_1+S_1\sigma_2$	
Word 4	S_4	$S_4 + S_3\sigma_1$	$S_4 +S_3\sigma_1+S_2\sigma_2$	
Word 5	S_5	$S_5 + S_4\sigma_1$	$S_5 +S_4\sigma_1+S_3\sigma_2$	
Word 6	S_6	$S_6 + S_5\sigma_1$	$S_6 +S_5\sigma_1+S_4\sigma_2$	
Word 7	S_7	$S_7 + S_6\sigma_1$	$S_7 +S_6\sigma_1+S_5\sigma_2$	
Word 8	S_8	$S_8 + S_7\sigma_1$	$S_8 +S_7\sigma_1+S_6\sigma_2$	
Word 9	S_9	$S_9 + S_8\sigma_1$	$S_9 +S_8\sigma_1+S_7\sigma_2$	
Word 10	S_{10}	$S_{10}+ S_9\sigma_1$	$S_{10}+S_9\sigma_1+S_8\sigma_2$	
Word 11	S_{11}	$S_{11}+ S_{10}\sigma_1$	$S_{11}+S_{10}\sigma_1+S_9\sigma_2$	
Word 12	S_{12}	$S_{12}+ S_{11}\sigma_1$	$S_{12}+S_{11}\sigma_1+S_{10}\sigma_2$	
Word 13	S_{13}	$S_{13}+ S_{12}\sigma_1$	$S_{13}+S_{12}\sigma_1+S_{11}\sigma_2$	
Word 14	S_{14}	$S_{14}+ S_{13}\sigma_1$	$S_{14}+S_{13}\sigma_1+S_{12}\sigma_2$	
Word 15	S_{15}	$S_{15}+ S_{14}\sigma_1$	$S_{15}+S_{14}\sigma_1+S_{13}\sigma_2$	
Word 16	S_{16}	$S_{16}+ S_{15}\sigma_1$	$S_{16}+S_{15}\sigma_1+S_{14}\sigma_2$	

TABLE 4.7

until S_{15} is multiplied by σ_1 and added to S_{16}. The subcripts of the terms multiplied together always sum to the syndrome number to which they are added. For example, fifteen plus one equals sixteen. The addition is Galois addition (exclusive or) and the output is in code form.

In the next pass σ_2 is added to $S_2 + S_1 \sigma_1$ in word 2. Again the sum of the subcripts in each term of the coefficient equals the subcript of the syndrome to which they are added. This sum also equals the word number in the output table TMSYN. This provides an easy identifying relationship between the σ coefficients and the next starting point in table TMSYN. Word 2 now contains $S_2 + S_1\sigma_1 + \sigma_2$. The coefficient σ_2 is now multiplied by S_1 and added to the contents of word 3. This process continues until S_{14} is multiplied by σ_2 and is added to the contents of word 16.

Each subsequent pass will take the next σ coefficient and add it to the contents of a word in table TMSYN which is one higher than the starting point of the previous pass. The final pass will be the addition of σ_s to word s where s is the number of erasures.

The first s terms of Table TMSYN (s equals the number of erasures) will be used later by subprogram RPGEN to compute the errata polynomial. Refer to section 4.7. The remaining 16-s terms are the modified syndromes. Word 0 of table TMSYN contains the number of erasures , the value s . Word 0 will be used by subprogram EPGEN as a pointer to the position in Table TMSYN of the first modified syndrome. Refer to section 4.5. Word 0 will also be used by subprogram RPGEN as an indication of which coefficients of this product of the T_j terms should be included in the generation of the errata polynomial. Refer to section 4.7.

MSYNG converts the values stored in code form in Table TMSYN to power form and stores them in Table TMSYP. This makes the power form of the results available to both subprogram EPGEN and RPGEN. Word 0 of Table TMSYP contains the number of erasures.

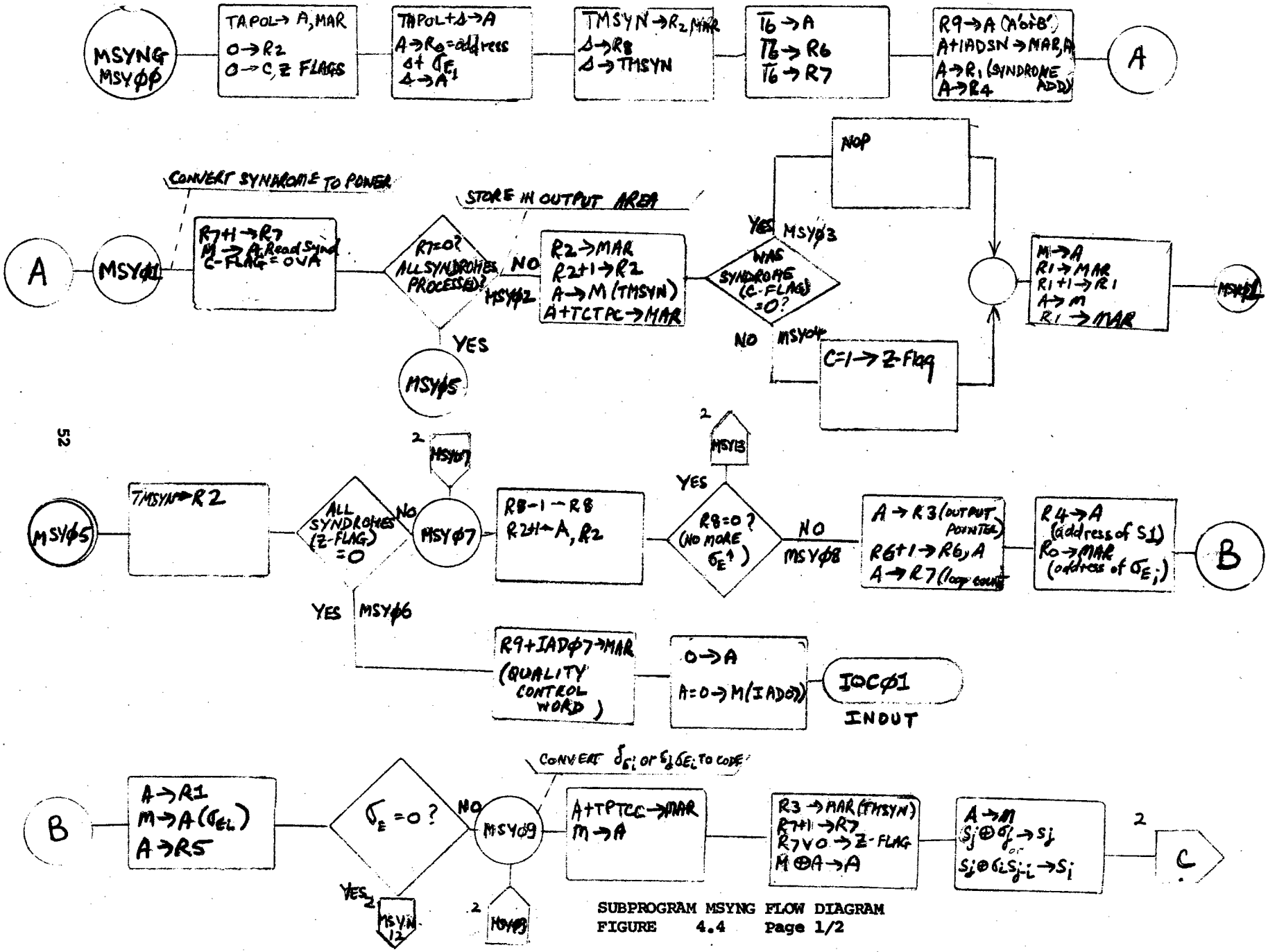
4.4.2 Flow Diagram

Figure 4.4 is the flow diagram for subprogram MSYNG.

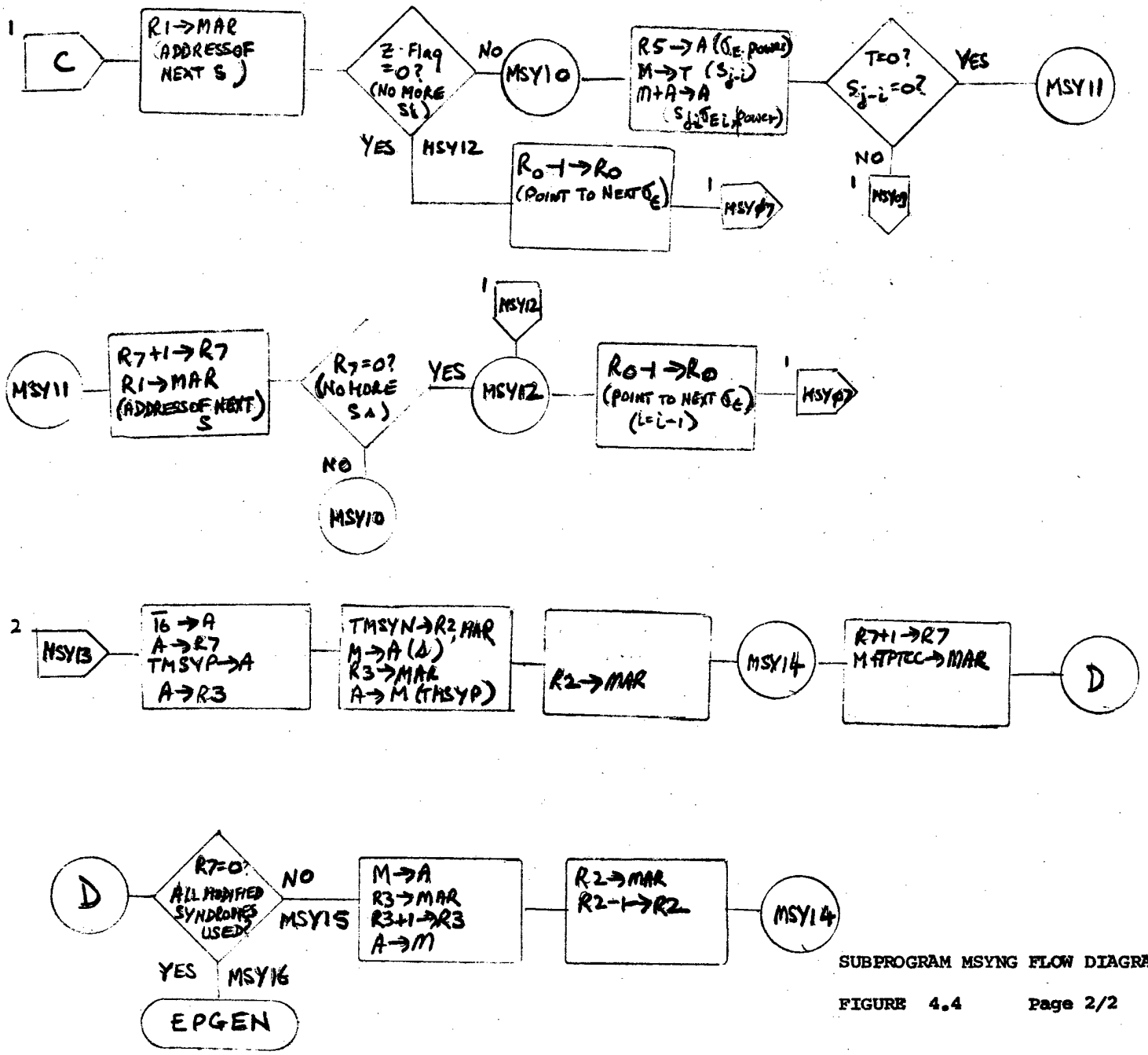
4.4.3 Environment

4.4.3.1 Tables

MSYNG uses only common data base tables.



SUBPROGRAM MSYNG FLOW DIAGRAM
FIGURE 4.4 Page 1/2



SUBPROGRAM MSYNG FLOW DIAGRAM

FIGURE 4.4 Page 2/2

4.4.3.2 Variables

MSYNG uses only common data base variables.

4.4.3.3 Constants

MSYNG does not use any constants other than the use of the K bus which is defined in the appropriate instruction.

4.4.3.4 Flags

MSYNG uses only common data base flags.

4.4.3.5 Indexes

Table 4.8 describes the indexes used by MSYNG.

4.4.3.6 Common Data Base Reference

The following data base items are referenced by MSYNG:

ITEM	TABLE
IADSN + 0 to + 15	TADIO
IBDSN + 0 to 15	TBDIO
IAD07	TADIO
IBD07	TBDIO
TCTPC	TCTPC
TPTCC	TPTCC
TAPOL + 0 to + 16	TAPOL
TMSYN + 0 to + 16	TMYSN
TMSYP + 0 to + 16	TMSYP
IADES	TADIO
IBDES	TBDIO

CPE REGISTERS USED AS INDEXES BY : MSYNG

REGISTER	CONTENT/PURPOSE
R 0	Address pointer to table TAPOL.Initially set to TAPOL + s. It is decremented by 1 for each σ coefficient used.
R 1	Address pointer to syndromes in data block. It is reset each time a new σ coefficient is used.
R 2	Address pointer when syndromes are transferred to TMSYN. Address pointer to TMSYN as partial outputs are computed when it requires to be reset on each pass.
R 3	Initial Address pointer for TMSYN to be used to reset R_2.
R 4	Address of the first syndrome S_1 i.e. IADSN/IBDSN.
R 5	Current σ_i in power form.
R 6	Number of output words to be modified this pass in one's complement form. It is incremented for each new σ coefficient.
R 7	1. Counter for loop which transfers syndromes from input area to TMSYN and converts syndromes in the input area to power form. 2. Inner loop counter, Set to R_6 when new σ is used.
R 8	The number of erasures i.e. s. Decremented each pass and is used to terminate computation.
R 9	Current input/output data block indicator. Equals 0 when A is current Data Block Equals 200(Ø)when B is current Data Block
T	Used to test if syndromes are zero.

TABLE 4.8

4.4.4 Input/Output Formats

Inputs and outputs processed by MYSNG are listed below together with their common data base references which describe them in detail:

ITEM	INPUT	OUTPUT	REFERENCE
IADSN + 0 to + 15	X		TADIO
IBDSN + 0 to + 15	X		TBDIO
IAD07		X	TADIO
IBD07		X	TBDIO
TCTPC	X		TCTPC
TPTCC	X		TPTCC
TAPOL + 0 to + 16	X		TAPOL
TMSYN + 0 to + 16		X	TMSYN
TMSYP + 0 to + 16		X	TMSYP
R_9	X		Table 4.8
IADES	X		TADIO
IBDES	X		TBDIO

4.4.5 Conditions for Initiation

APGEN passes control to MSYNG when it has completed its own functions. MSYNG is always entered as part of the decoding process unless APGEN has made an abnormal decode failure exit to INOUT.

4.4.6 Limitations

MSYNG does not have any special limitations.

4.4.7 Interface Description

Reference Figure 3.3 for a block diagram showing the sequential and functional relationship of all subprograms.

MSYNG computes the modified syndromes and then passes control to subprogram EPGEN. If MSYNG determines that the R/S code word has no errata it makes an abnormal exit to INOUT at entry point IOC01. Refer to section 4.2. Control is then returned to subprogram INOUT.

4.5 Subprogram EPGEN

4.5.1 Detailed Description

The function of subprogram EPGEN is to compute the error polynomial. The error polynomial is defined as:

$$\sigma^{(u)}(X) = \sum_{i=0}^{n} \sigma_i^{(u)} X^i$$

where
u = current iteration, u = -1, 0, 1, ...(16-s)
s = number of erasures

The generation of the error polynomial is an iterative process, successive iterations depending on the results of previous iterations. The (u+1)th iteration is defined as:

Equation 1

$$\sigma^{(u+1)}(X) = \sigma^{(u)}(X) + du\, d_{\rho}\, X^{u-\rho}\, \sigma^{(\rho)}(X)$$

where
ρ = a selected previous iteration
du= a coefficient index, GF32 element
lu= a decimal computation number
u-lu = a decimal computation number

Initial iterations are defined as follows:

Iteration u	Polynomial $\sigma^{(u)}(X)$	du	lu	u-lu
-1	1	1	0	-1
0	1	T_1	0	0

where
T_m = the modified syndromes
and
m = 1,2, ...(16-s)

Using these initial iterations, suceeding iterations can be computed using Equation 1 (defined above) and Equations 2 and 3 (defined below) .

Equation 2

$$du = \sum_{\forall\, m+n = u+1} T_m \sigma_n^{(u)}$$

$$= T_{u+1}\sigma_0^{(u)} + \ldots\ldots\ldots\ldots + T_0\sigma_{u+1}^{(u)}$$

where σ s are the coefficients of $\sigma^{(u)}(X)$

and

if $du = 0$, then $\sigma^{u+1}(X) = \sigma^{u}(X)$ and $l_{u+1} = l_u$

Equation 3

When $du \neq 0$, ρ is a previous iteration selected so that

$$d\rho \neq 0$$

$$l_{u+1} = \text{maximum of } l_u \text{ and } l_\rho + (u-\rho)$$

EPGEN begins its processing by checking the number of erasures. If 16 erasures exist, then there will be no error polynomial and so an exit will be made at once to RPVAL at entry point RPV00.

If no exit is made, EPGEN initializes CPE registers R_6, R_7 and R_8 as pointers to the three output areas TEPAA, TEPBB, TEPCC. These tables will be used to store the coefficients of $\sigma(X)$, the error polynomial, for the ρ, u and u+1 iterations. EPGEN uses one other table for intermediate storage, table TEPDR. This is used to store the values d , du, ρ , $l\rho$, u, and lu which are required for the computation of the u+1 th terms.

The modified syndromes in code form in TMSYN and in power form in TMSYP are input for EPGEN. EPGEN reads the first modified syndrome and this becomes the first du term. It is checked for zero value to see if the u+1 th terms have to be calculated. If the value is zero, the u+1 th terms equal the u th terms and the next modified syndrome will be selected.

If the value of du is not zero, then EPGEN computes the term $l_\rho + (u-\rho)$, reference Equation 3. EPGEN then copies $(u - \rho - 1)$ terms from the u storage to the $(u+1)$ th storage area, reference equation 1. This copies $\sigma^{(u)}(X)$ into the area for $\sigma^{(u+1)}(X)$. The remaining terms of equation 1 i.e.

$$du \; d_\rho^{-1} \; X^{u-\rho} \; \sigma^{(\rho)}(X)$$

are now computed. The initial values of ρ is -1, of u is o, of du is T_1 (the first modified syndrome), of d_ρ is 1 and of $\sigma^{(\rho)}(X)$ is 1. Refence the table above.

EPGEN now checks to see if the previous ρ iteration should be changed for the next iteration. This is done by computing $l(u)$ and $(\rho) + u - \rho$. A test is then made and if

$$l(u) - u > l(\rho) - \rho$$

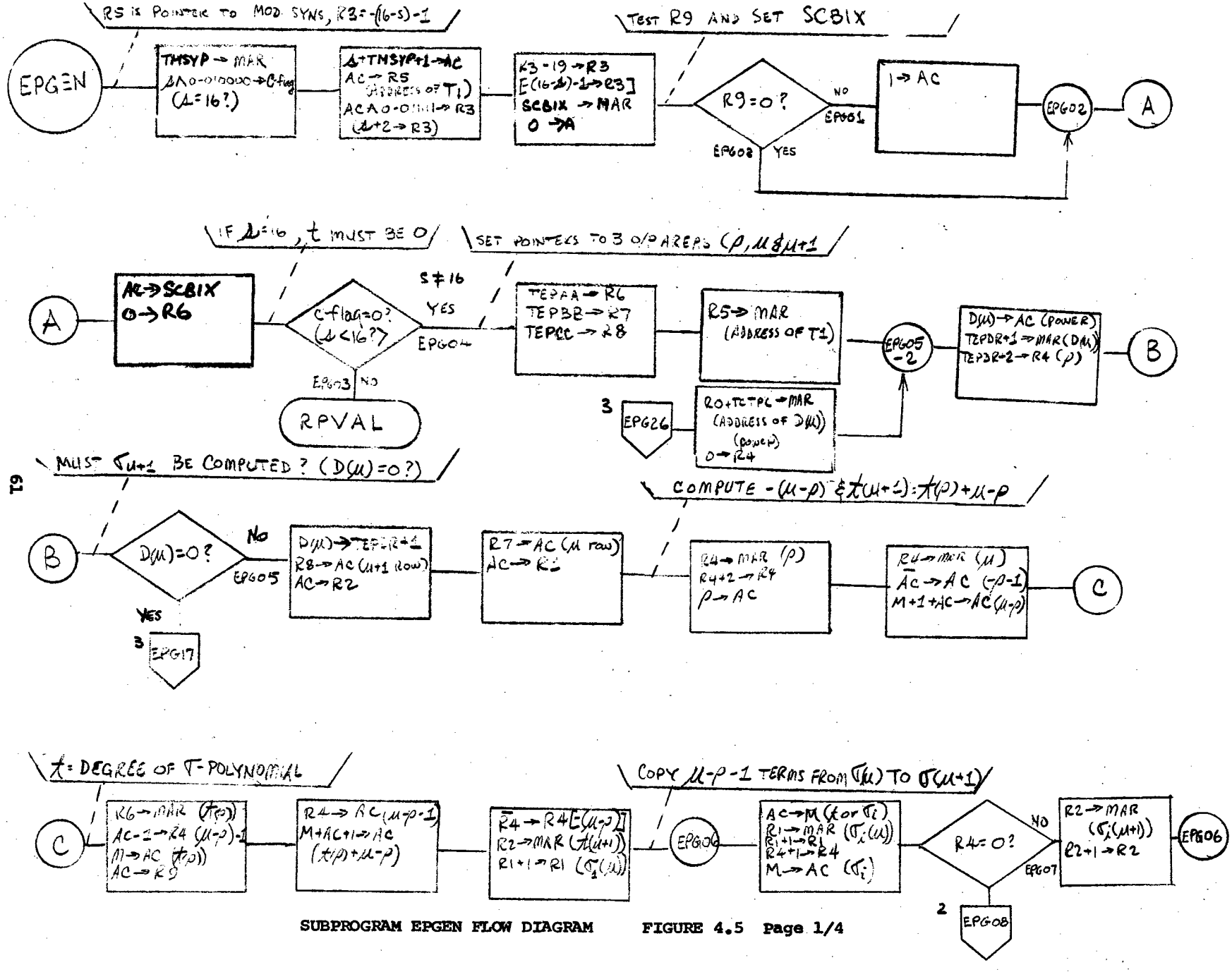
then the terms in ρ storage are replaced by those in the u storage and the address pointers for ρ and u are exchanged.

The first iteration has now been completed. Address pointers are now exchanged for $\sigma(u)$ and $\sigma(u+1)$ since the $(u+1)$ terms now become the u th terms. The next du term is computed using equation 2. The modified syndrome one number higher then the current du number is selected and to it is added the product of the next lower numbered modified syndrome and the first coefficient of $\sigma(u)$. As this computation progresses in further iterations it is seen that the product pairs selected always equal one more than the u value being computed. When du has been computed it is tested for zero value as before. If it is not zero, then the next iteration of $\sigma(u+1)$ is computed.

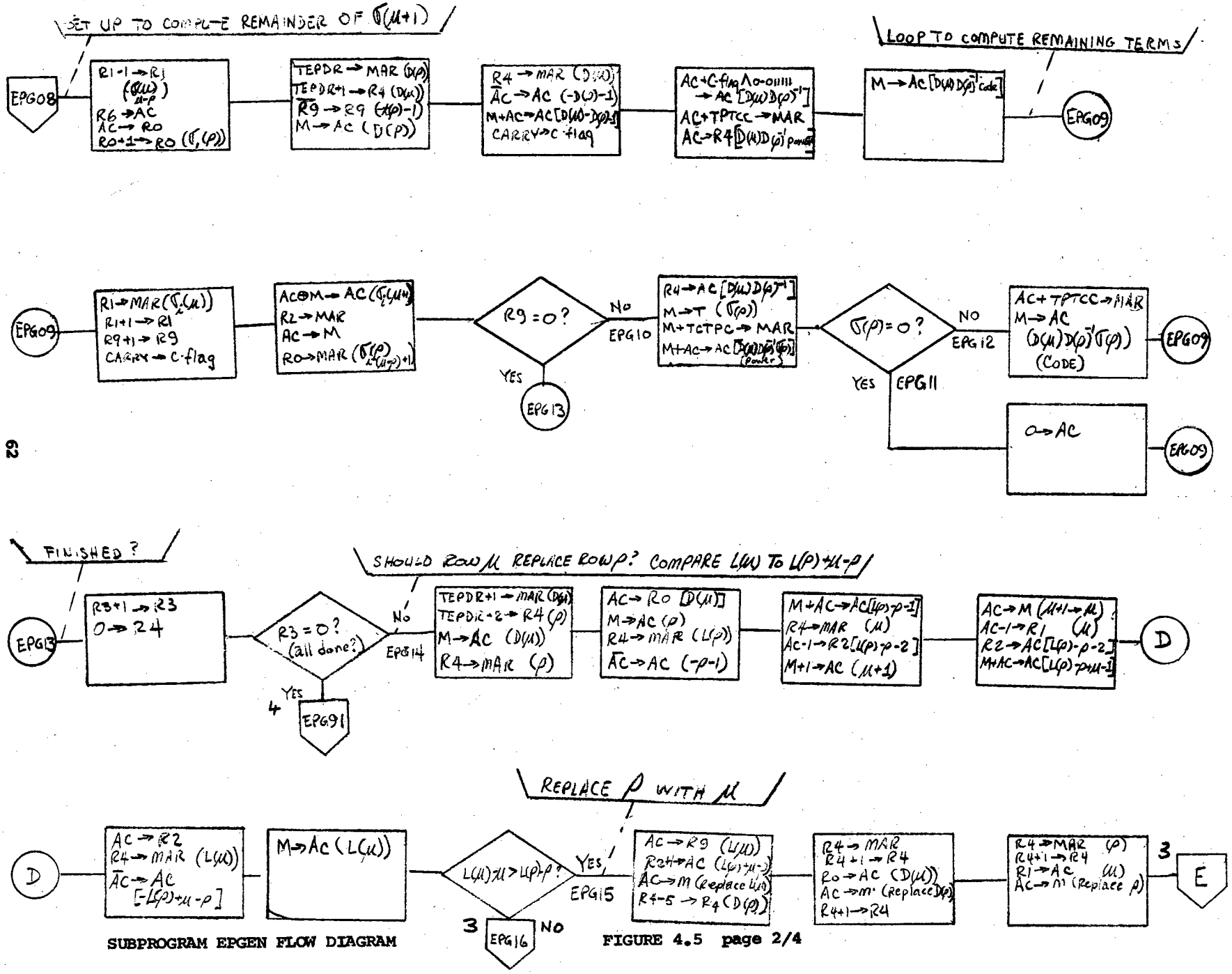
When 16-s, where s is the number of erasures, iterations are computed the last $\sigma(u+1)$ is the error polynomial. EPGEN then checks the degree (the number of coefficients found) of the polynomial. If the degree is zero, an exit is made to subprogram RPVAL to bypass the remaining error processing. If the degree is such that $2e + E > 16$, then a decode failure is declared and an exit is made to INOUT at entry point IOC01. If neither of these conditions exist, EPGEN restores the value of the current block indicator in R_9 and then passes control to subprogram EPVAL.

4.5.2 Flow Diagram

Figure 4.5 is the flow diagram for subprogram EPGEN

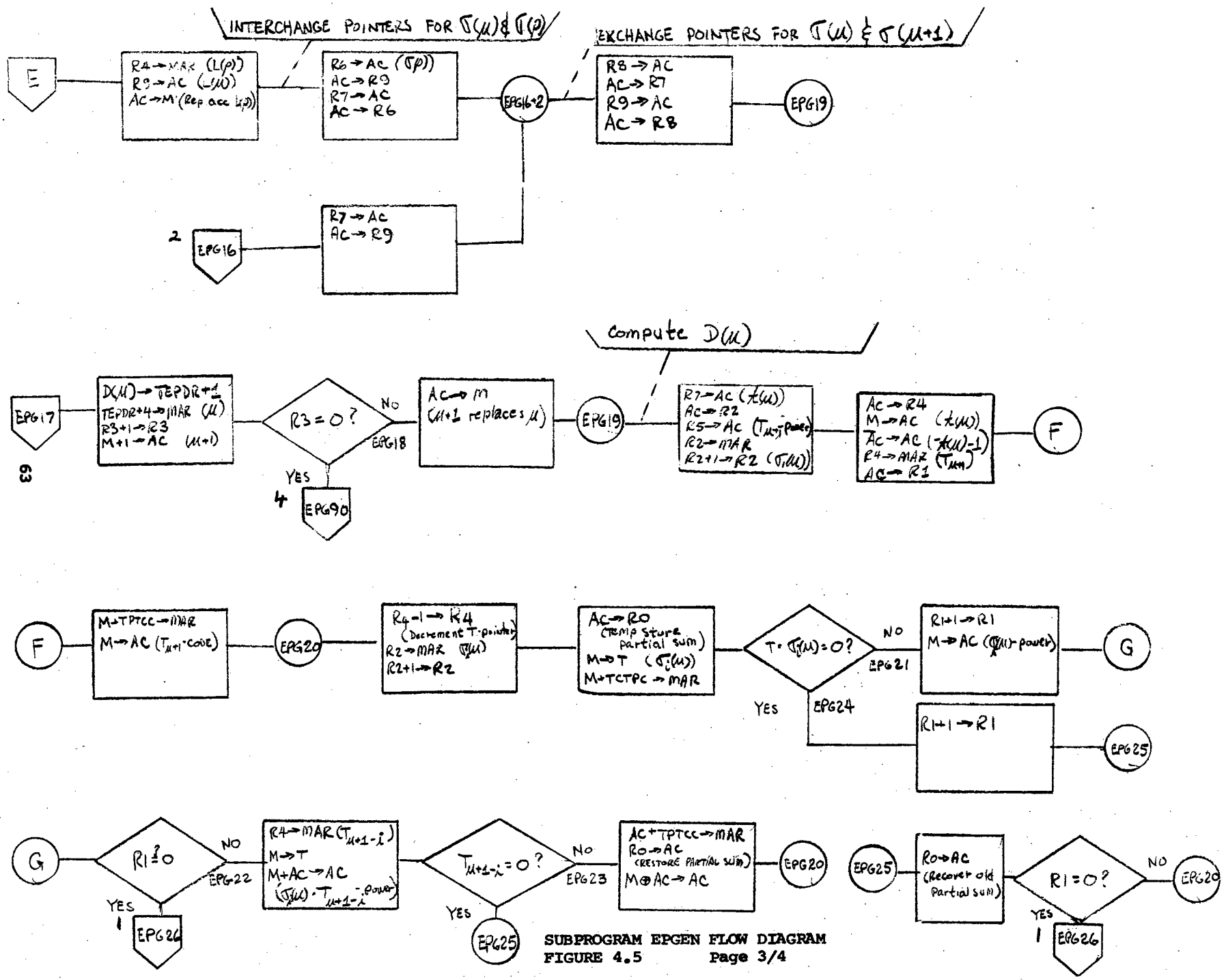


SUBPROGRAM EPGEN FLOW DIAGRAM FIGURE 4.5 Page 1/4

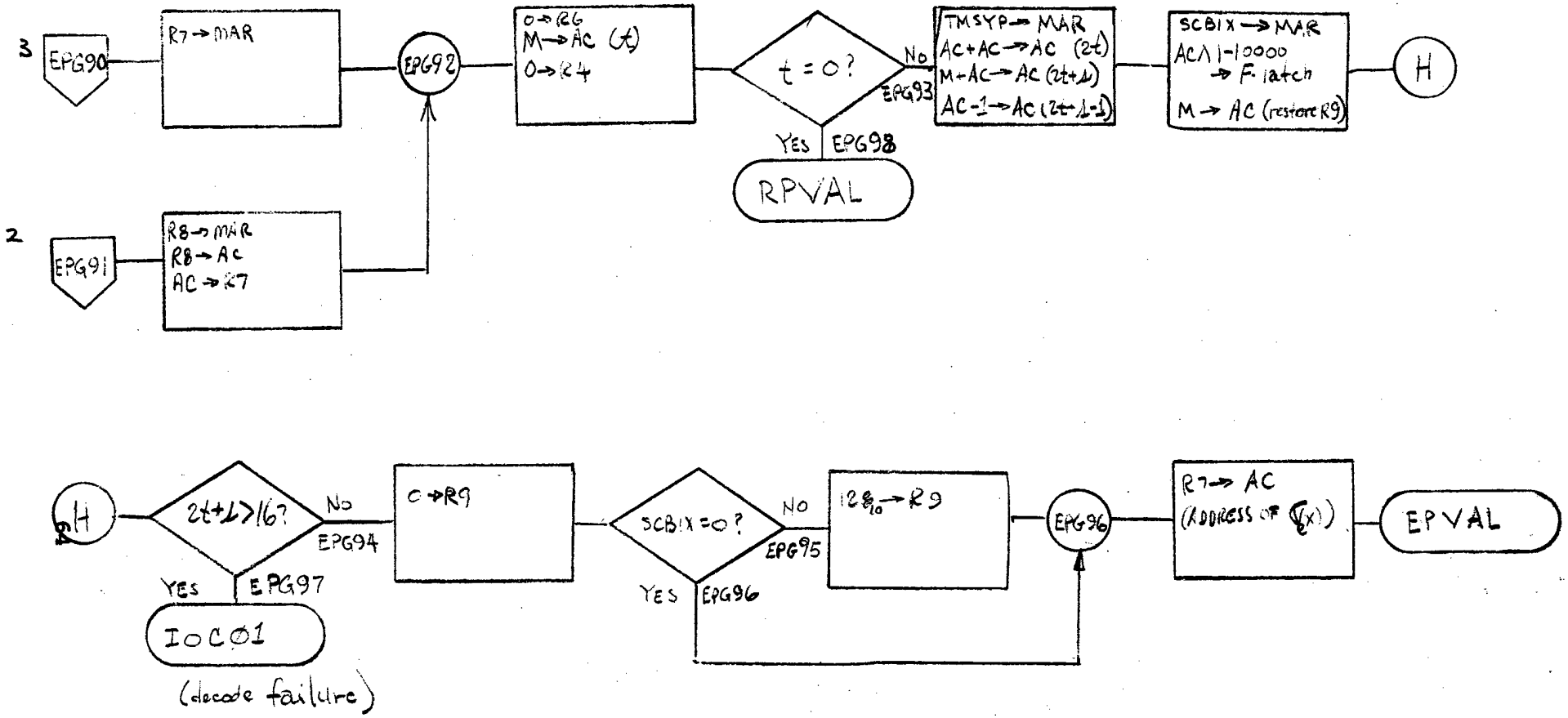


SUBPROGRAM EPGEN FLOW DIAGRAM

FIGURE 4.5 page 2/4



SUBPROGRAM EPGEN FLOW DIAGRAM
FIGURE 4.5 **Page 3/4**



SUBPROGRAM EPGEN FLOW DIAGRAM

FIGURE 4.5 Page 4/4

4.5.3 Environment

4.5.3.1 Tables

The following table is used only by EPGEN

- o Table name is TEPDR
- o Purpose

  TEPDR is used by EPGEN to store the values $d\rho$, du, ρ, 1ρ, u and lu. These are terms which are computed on each iteration by EPGEN and are stored in TEPDR for intermediate storage.TEPDR is initialized by subprogram INOUT as a time saving device . The initialization is done before each new R/S code word is received. Reference Table 4.9

- o Size and Indexing

  TEPDR is a table 6 words long(fixed lenghh, vertical type). It is addressed by using the K bus instruction setting to be the data/Ram address of word 0 i.e. 733(Ø). A CPE index register is set to this value and is automatically incremented by one whenever the table is written into or read from. This provides the address of the next sequential word in the table.

- o Structure and Bit Layout

  Table 4.9 shows the format of TEPDR.

4.5.3.2 Variables

EPGEN accesses only common data base variables.

4.5.3.3 Constants

EPGEN does not use any constants other than the use of the K bus which is defined in the appropriate instruction.

4.5.3.4 Flags

EPGEN uses only common data base flags.

4.5.3.5 Indexes

Table 4.10 describes the indexes used by EPGEN.

TABLE NAME: TEPDR

RAM ADDRESS	WORD NUMBER	BIT LOCATIONS 7 6 5 4 3 2 1 0	CONTENTS
OCTAL			
733	00	X X X	d term computed on each iteration
734	01	X X X	du "
735	02	X X X	"
736	03	X X X	l "
737	04	X X X	u "
740	05	X X X	lu "

INITIAL SETTINGS ARE AS FOLLOWS:

733	037
734	000
735	000
736	000
737	001
740	000

TABLE 4.9

CPE REGISTERS USED AS INDEXES BY : EPGEN

REGISTER	CONTENT/PURPOSE
R 0	
R 1	Used for the u th iteration terms.
R 2	Used for the (u+1) th iteration terms.
R 3	The iteration counter. It contains the one's complement of the number of modified syndromes i.e. (16-s)
R 4	Used for intermediate storage of terms being computed.
R 5	Address of the next modified syndrome.
R 6	Address of the next word of table TEPAA
R 7	Address of the next word of table TEPBB
R 8	Address of the next word of table TEPCC
R 9	Used to hold computation results . Finally is restored to the current data block indicator. Equals 0 when A is current data block Equals 200(O)when B is current data block
T	

TABLE 4.10

4.5.3.6 Common Data Base Reference

The following data base items are referenced by EPGEN:

ITEMS	TABLE
SCBIX	SCBIX
TMSYN + 0 to + 16	TMSYN
TMSYP + 0 to + 16	TNSYP
TCTPC	TCTPC
TPTCC	TPTCC

4.5.4 Input/Output Formats

Inputs and outputs processed by EPGEN are listed below together with their common data base references which describe them in detail:

ITEM	INPUT	OUTPUT	REFERENCE
SCBIX	X		SCBIX
TMSYN + 0 to + 16	X		TMSYN
TMSYP + 0 to + 16	X		TMSYP
TCTPC	X		TCTPC
TPTCC	X		TPTCC
R_9	X	X	Table 4.10
TEPOL= TEPAA		X	TEPOL
TEPBB		X	
TEPCC		X	

4.5.5 Conditions for Initiation

Control is always passed to EPGEN by MSYNG as part of the normal decoding process unless MSYNG has detected that the R/S code word has neither errors or erasures.

4.5.6 Limitations

EPGEN does not have any special limitations.

4.5.7 Interface Description

Reference Figure 3.3 for a block diagram showing the sequential and functional relationship of all subprograms.

EPGEN computes the coefficients of the error polynomial and then passes control to subprogram EPVAL exiting to EPV00. This is the normal processing exit. If EPGEN detects that the error polynomial has zero coefficients or that there are 16 erasures in the R/S code word, it passes control to subprogram RPVAL to bypass the remaining error processing. Entry is at RPV00.If EPGEN detects the decode failure of $2e + E > 16$, it returns control to INOUT at entry point IOC01.

4.6 Subprogram EPVAL

4.6.1 Detailed Description

The major function of subprogram EPVAL is to solve for the roots of the error location polynomial. The error location polynomial is defined as follows:

$$\sigma_e(X) = \sum_{n=0}^{t} \sigma_{e_n} (X)^n$$

If the roots of the polynomial are defined as

$$\alpha^j, \text{ where } j = 0,1,.....t$$

then

$$\sum_{n=0}^{t} \sigma_{e_n} (\alpha^j)^n = 0 \text{ for all values of } j.$$

EPVAL determines which values of α cause the polynomial expression to reduce to zero. Subprogram EPGEN has computed the coefficients of the error location polynomial and stored them in table TEPOL. Table TEPOL is the table in which the final output of EPGEN has been stored. It is always in one of three possible areas in the Data Ram i.e. in table TEPAA, or in TEPBB, or in TEPCC. Reference section 4.5 for a description of EPGEN output. EPGEN stores the variable output address in the accumulator and then passes control to EPVAL. The first function performed by EPVAL is to store the address of the error location polynomial in the Data Ram in SEPEP. EPVAL then reads the number of coefficients in the error location polynomial. This will be greater than one and less than seventeen. Reference EPGEN functions in 4.5. EPVAL checks to see if there are either one or two coefficients. If there are ,two separate processing paths are selected. These are described later.

If there are more than two coefficients the main processing path is followed. First the error location polynomial coefficients are converted from code form and stored in power form in table TEVOA. Both the code and power form of these coefficients will be required. If there are t coefficients then there will be t roots if the decode is to be successful. The solution for the first t-2 roots is found by substituting values of α^j, where j runs from 1 to 31. The computation is done in the form of a row column matrix where the 31 α terms are the row terms. The number of coefficients t are the column terms. The maximum value of t is 8. Therefore the maximum matrix size is 31 by 8.

The first row to be computed is as follows:

$$\text{Row Sum}_1 = 1 + \sigma_1 \alpha^1 + \sigma_2 (\alpha^1)^2 + \sigma_3 (\alpha^1)^3 \ldots\ldots + \sigma_t (\alpha^1)^t$$

The row term for the first row is α^1. The column term is set equal to the row term to begin i.e. the first column term is α^1.

The column term is multiplied by the first coefficient to give $\sigma_1 \alpha^1$.

This is added to the partial Row Sum which is always initialized to 1. The partial Row Sum is then

$$\text{Row Sum}_1 = 1 + \sigma_1 \alpha^1$$

The current column term α^1 is then multiplied by the current row term α^1 to give the new column term $(\alpha^1)^2$.

The new column term is now the current term and so is multiplied by the next coefficient and added to the partial Row Sum to give

$$\text{Row Sum}_1 = 1 + \sigma_1 \alpha^1 + \sigma_2 (\alpha^1)^2$$

The current column term is now multiplied by the row term to give the new current column term $(\alpha^1)^3$.

The process continues until the last coefficient σ_t has been multiplied by the current column term $(\alpha^1)^t$, to give $\sigma_t (\alpha^1)^t$ and added to Row Sum_1.

Row Sum_1 is then checked for zero value and if it equals zero, a root has been found. When a root is found, the inverse value defined as

β^{31-n} where α^n is a root

is stored in table TEVAL. The inverse root is cummulatively added into the cummulative sum SIRCS. (Galois add equals exclusive or.) The inverse root is also added to the cummulative product SIRPS. (Galois product equals add.) A check is now made to see if t-2 roots have been found. If they have the processing continues with the method for solving for two roots, described later.

If t-2 roots have not been found or if no root was found (Row Sum_1 not equal to zero), then the row term is incremented by 1 giving the next row term of α^2.

The new Row Sum_2 is initialized to the value 1 , the column term is set equal to the new row term , the first coefficient is multiplied by the column term and added to Row Sum_2 . The next column term is formed by multiplying the current column term by the row term. This is then multiplied by the next coefficient. The process continues until Row Sum_2

has been computed as

$$\text{Row Sum}_2 = 1 + \sigma_1(\alpha^2) + \sigma_2(\alpha^2)^2 + \sigma_3(\alpha^2)^3 + \ldots\ldots + \sigma_t(\alpha^2)^t$$

Row Sum_2 is then checked for zero as a possible root.

Processing continues in this way until all rows have been checked and either insufficient roots have been found or until t-2 roots have been found. During the process described above, a check is made to see if the inverse of the next α term selected has a value at which location an erasure is known to exist. If this is the case, the substitution of this α value need not be performed since by definition an error location will not be the same as an erasure location.

If the substitution process has not found t-2 roots then a decode failure is declared and EPVAL exits to INOUT at entry point IOC01 and control is given back to INOUT.

If t-2 roots have been found , two roots remain to be determined. The processing paths of the case of two error location coefficients and therefore two roots, and the main processing path now join. The solution for two roots makes use of the relationship between the coefficients of a quadratic equation and its roots namely:

the equation

$$1 + a X + b X^2$$

has roots such that the product of inverse of roots = b

the sum of the inverse of roots = a

The inverse roots cummulative product SIRPS is subtracted from the maximum σ_t (highest power coefficient). If a negative number results, 31 is added to the result , converting it to a positive number by the rules of Galois field arithmetic. This value is called p. Two numbers have to be found whose Galois product equals p, and whose Galois sum together with the inverse roots cummulative sum SIRCS, equals the lowest order coefficient σ_1 .

Pairs of numbers whose Galois product equals p will be the set of numbers which add to either p or p+31. Making use of the Galois field cycle of 31 for this R/S code , for example

$$\alpha^{43} = \alpha^{31} \cdot \alpha^{12} = \alpha^{12}$$

the full range of pairs of values adding to either p or p+31 need not be tried for possible root solutions. In addition, in the case of a short code, there are other limitations on the possible locations of errors. Error positions for a sucessful decode may only be located within the data or parity characters excluding the eleven leading zeros and the four trailing erasures from the range to be searched.

The range of values searched is as follows:

CODE TYPE		RANGE UPPER LIMIT	RANGE LOWER LIMIT
LONG		p - 1	1
	and	31	p
SHORT	for $p \geq 9$ and $p-4 < 20$	p - 4	4
	for $p < 9$	19	p + 12
	for $p-4 \geq 20$	19	p - 19

The search is performed by taking the upper and lower limit values, which always sum to the value p, and exclusive or their code values together with the cummulative inverse roots code sum SIRCS. The result is checked for equality with the lowest order coefficient σ_1. If equality is found, a root pair has been found and the solution process is complete. If equality is not found then the next pair of trial values is found by incrementing the lower order limit by one and decrementing the upper limit by one. This leaves the sum of the pair still equal to p. The same process is now repeated. The search within a range terminates when the upper limit is no longer greater than the the lower limit. If there is another range to be searched this will be done in the same manner. Refer to the ranges listed above for long and short codes. As before if equality is found, a root pair has been found. If not, after the appropriate ranges have been searched a decode failure has occured and EPVAL exits to INOUT at entry point IOC01.

The above method for solving for the last two of a root set or for the case of two roots only, is used as a time saving device. In the main processing path, up to 31 α_k values (for a long code) can be substituted each on a trial root basis since errors may occur anywhere within the set of 31 characters. It depends upon the distribution of the errors position wise, as to how quickly the roots of the polynomial are found. The coding loop through which the trials for the quadratic solution pass is much shorter than the coding loop through which the row/column matrix trials pass.

The path for a one root solution, determined by the presence of one coefficient in the error location polynomial, is a separate path. EPVAL detects this case and the inverse root is the value of the coefficient itself.

For both the single root and two root (or last two roots) solutions, the values found are the inverses of the roots of the error location polynomial. EPVAL writes the power form of the value which,is the location of the error,in its output table TEVAL.

EPVAL completes its processing for a sucessful decode by storing the number of roots found in the first word of table TEVAL. The current data block indicator R_9 detroyed by EPVAL for its own processing reasons, is then restored . EPVAL then gives control to subprogram RPGEN as it exits to entry point RPG00.

4.6.2 Flow Diagram

Figure 4.5 is the flow diagram for subprogram EPVAL.

4.6.3 Environment

4.6.3.1 Tables

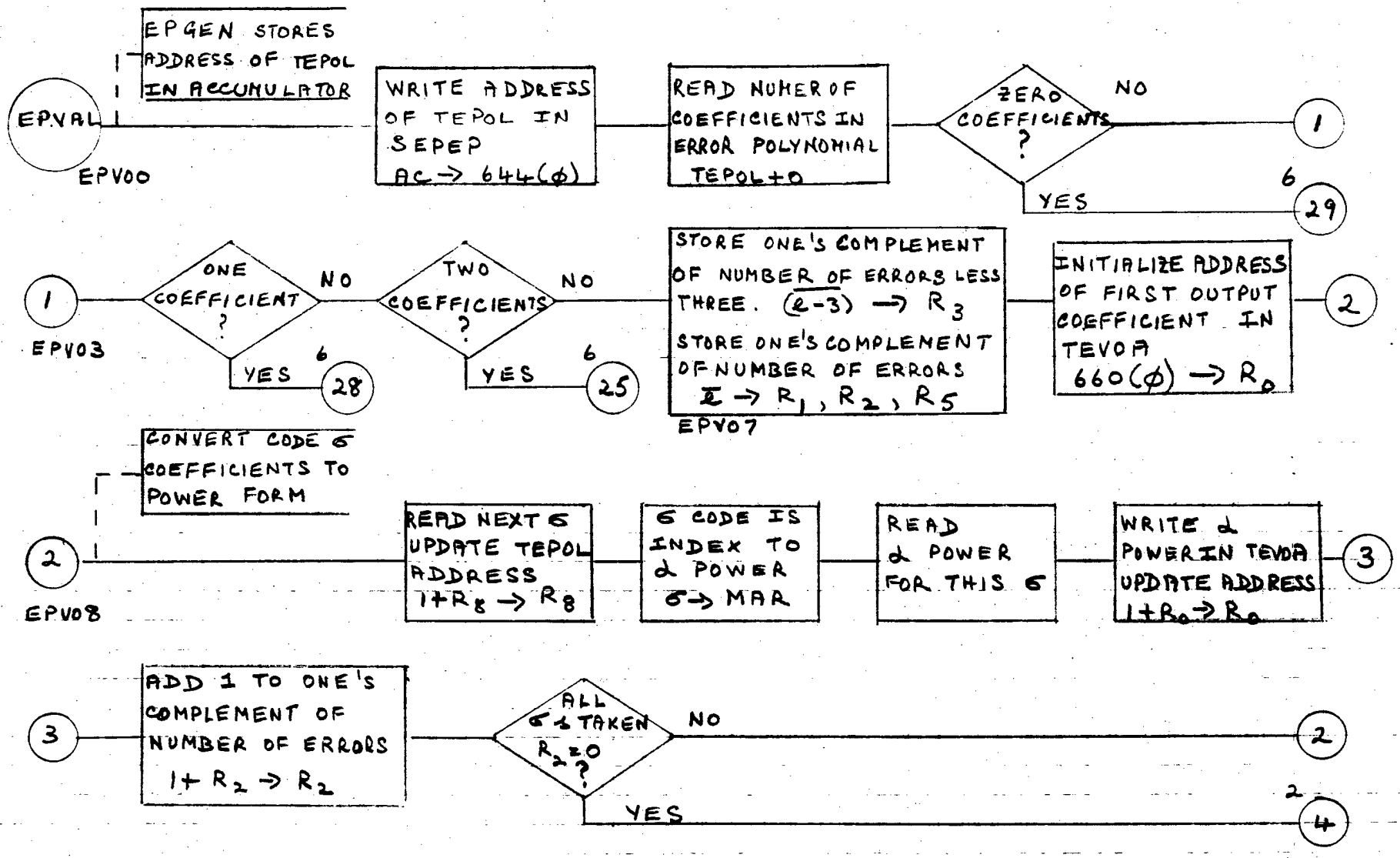
The following table is used only by EPVAL.

- Table name is TEVOA.

- Purpose .
  TEVOA is used by subprogram EPVAL to store the power form of the coefficients of the error location polynomial. Common table TEPOL , input to EPVAL, contains the code form of these coefficients.

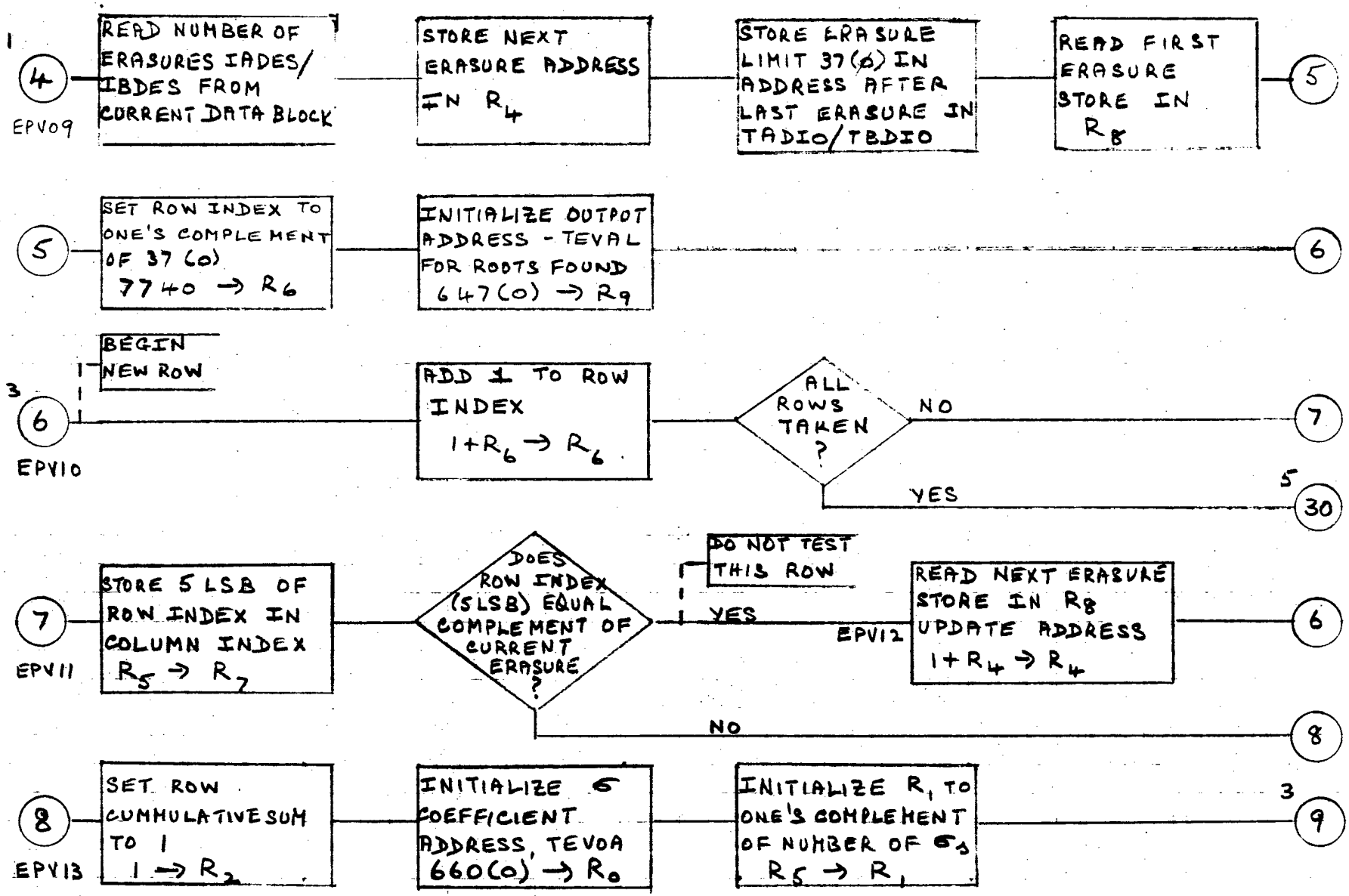
- Size and Indexing Procedure.
  TEVOA is a table of length 9 words (fixed length.)
  It is addressed by using the K bus instruction setting to be the Data Ram address of word 0 i.e. 657(Ø).
  A CPE index register is set to this value and is automatically incremented by one whenever the table is written into. This provides the address of the next sequential word in the table.

- Structure and Bit layout.
  TEVOA is a vertical type table.
  The table format is shown in Table 4.11



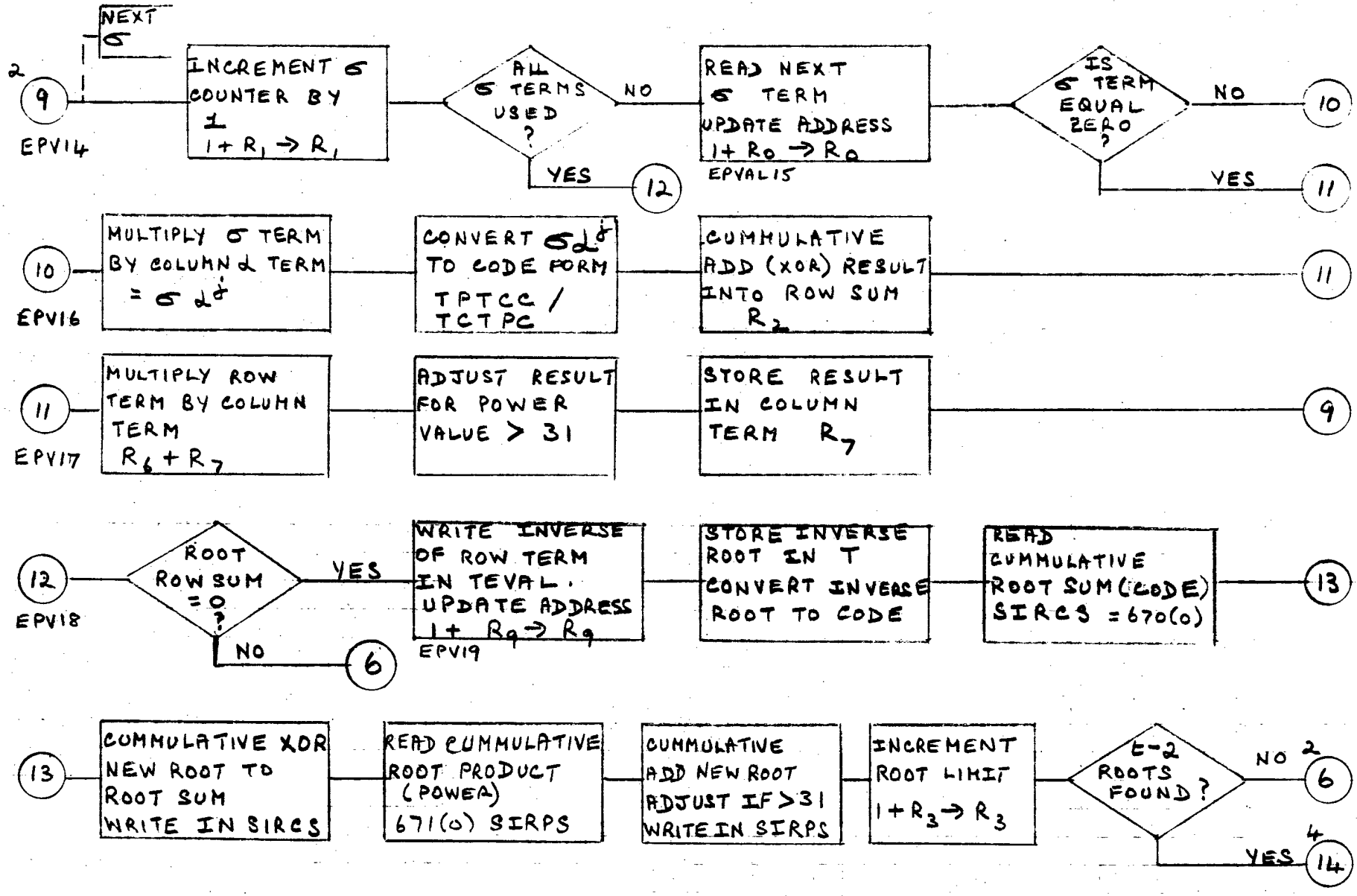
SUBPROGRAM EPVAL FLOW DIAGRAM

FIGURE 4.6 Page 1/6



SUBPROGRAM EPVAL FLOW DIAGRAM

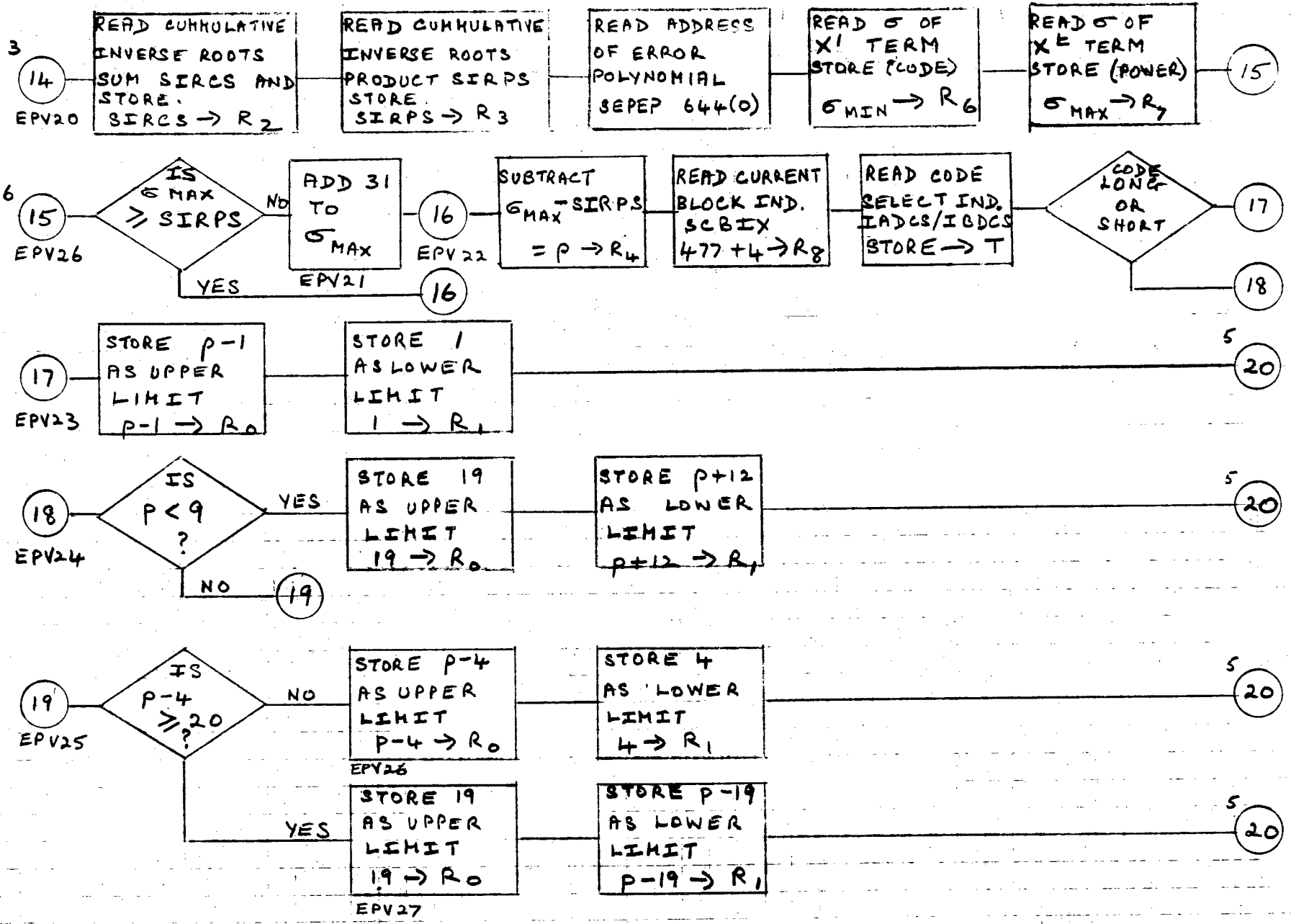
FIGURE 4.6 Page 2/6



77

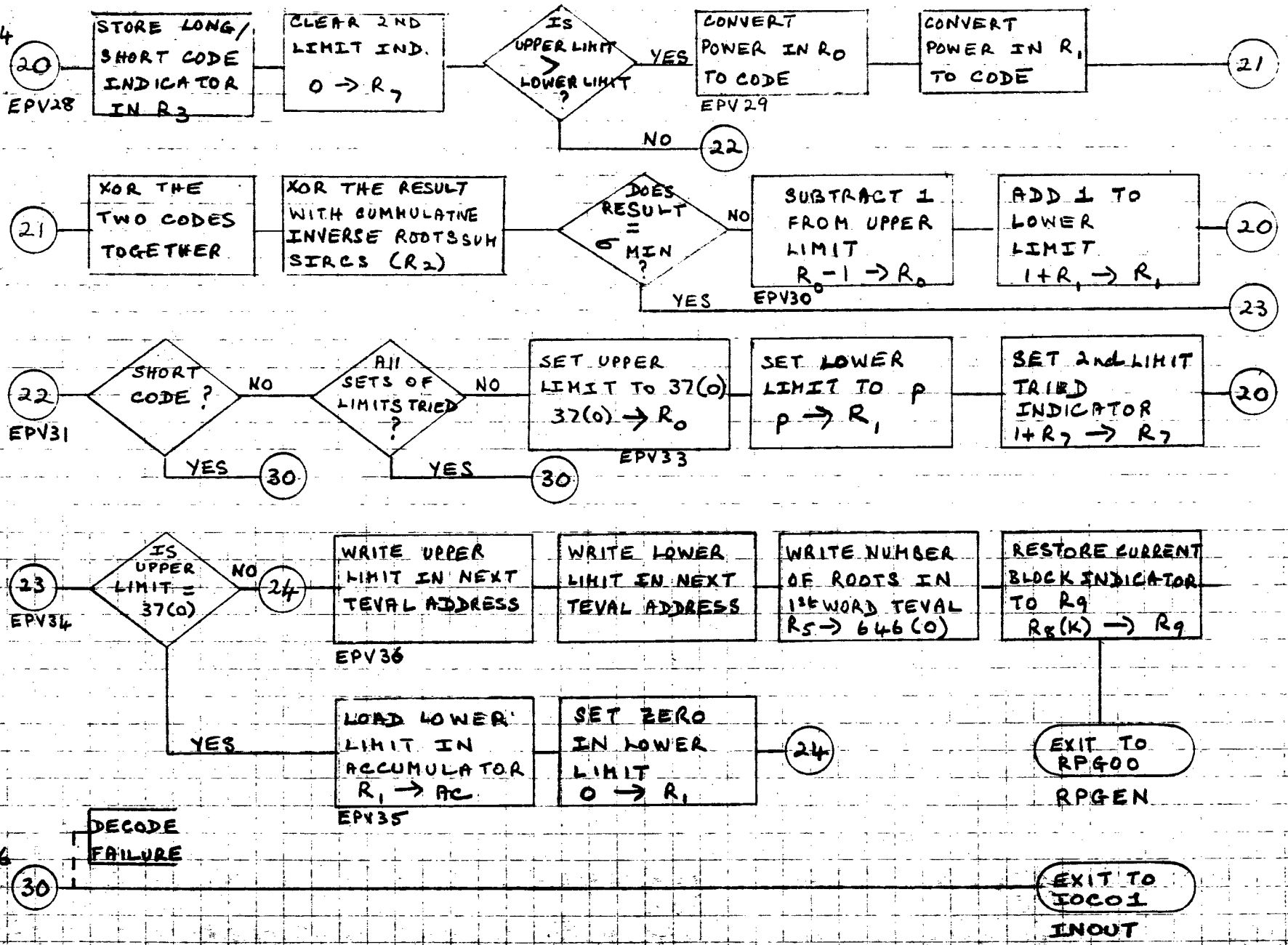
SUBPROGRAM EPVAL FLOW DIAGRAM

FIGURE 4.6 Page 3/6



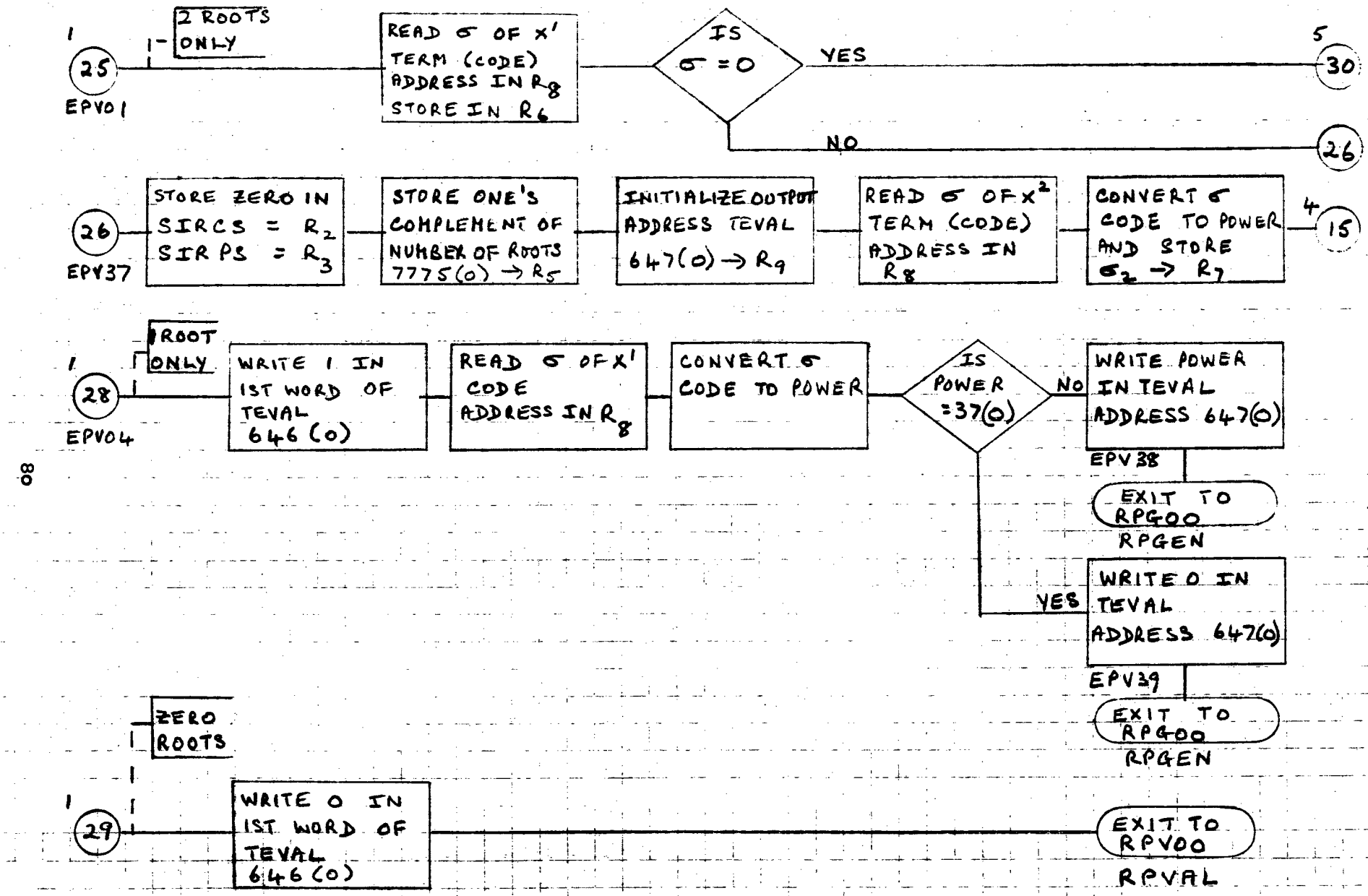
SUBPROGRAM FLOW DIAGRAM

FIGURE 4.6 Page 4/6



SUBPROGRAM EPVAL FLOW DIAGRAM

FIGURE 4.6 Page 5/6



SUBPROGRAM EPVAL FLOW DIAGRAM

FIGURE 4.6 Page 6/6

TABLE TYPE: INTERNAL TABLE NAME: TEVOA

RAM ADDRESS OCTAL	WORD NUMBER	BIT LOCATIONS 7 6 5 4 3 2 1 0	CONTENTS
657	00	X X X	Not used
660	01	X X X	Power form of first coefficient of error polynomial
661	02	X X X	.
662	03	X X X	.
663	04	X X X	.
664	05	X X X	.
665	06	X X X	.
666	07	X X X	.
667	08	X X X	Power form of highest coefficient of error polynomial

X = not used

TABLE 4.11

4.6.3.2 Variables

- SIRCS is a one word variable located at word 670(Ø) in the Data Ram. Only bits 4 through 0 are used.
  EPVAL stores the cummulative Galois sum (EXOR) of the inverse roots of the error polynomial in SIRCS in code form.
  SIRCS is initialized to zero by INOUT as a time saving device.

- SIRPS is a one word variable located at word 671(Ø) in the Data Ram. Only bits 4 through 0 are used.
  EPVAL stores the cummulative Galois product (addition) of the inverse roots of the error polynomial in SIRPS in power form.
  SIRPS is initialized to zero by INOUT as a time saving device.

4.6.3.3 Constants

EPVAL does not use any constants other than the use of the K bus which is defined in the appropriate instruction.

4.6.3.4 Flags

EPVAL uses only common data base flags.

4.6.3.5 Indexes

Table 4.12 describes the indexes used by EPVAL.

4.6.3.6 Common Data Base Reference

The following data base items are referenced by EPVAL:

ITEM	TABLE
IADCC/IBDCC	TADIO/TBDIO
IADCS/IBDCS	TADIO/TBDIO
IADEE + 0 to + 30	TADIO
IBDEE + 0 to + 30	TBDIO
IADES	TADIO
IBDES	TBDIO
SCBIX	SCBIX
SEPEP	SEPEP
TCTPC	TCTPC
TEPOL + 0 to + 8	TEPOL
TEVAL + 0 to + 8	TEVAL
TPTCC	TPTCC

CPE REGISTERS USED AS INDEXES BY : EPVAL

REGISTER	CONTENT/PURPOSE
R 0	Address for table TEVOA into which EPVAL writes the power form of the error location polynomial coefficients. Is incremented in the routine. Initially set to 660 (O).
R 1	The one's complement of the number of coefficients of the error location polynomial. Used to restore the setting of R_2.
R 2	The one's complement of the number of coefficients of the error location polynomial. It is incremented by 1 for each pass through the loop.
R 3	The one's complement of the number of coefficients of the error location polynomial less three.
R 4	
R 5	The one's complement of the number of coefficients of the error location polynomial. Is saved for use later in EPVAL.
R 6	
R 7	
R 8	Address of the next coefficient in input table TEPOL. Is incremented by 1 . Initially placed in the Accumulator by subprogram EPGEN.
R 9	Current data block indicator Equals 0 when A is current data block Equals 200(Ø)when B is current data block.
T	

CPE Registers as used by EPVAL during the first processing phase.

TABLE 4. 12 Page 1/3

CPE REGISTERS USED AS INDEXES BY : EPVAL

REGISTER	CONTENT/PURPOSE
R 0	Current address for next coefficient in table TEVOA. in power form (EPVAL output). Initially set to 660(Ø). Incremented by 1.
R 1	The one's complement of the number of coefficients of the error location polynomial. Is incremented by 1. It is reset to its initial value for each new row.
R 2	The cummulative sum of polynomial terms for the current row i.e. Row Sum. It is initialized each time a new λ substitution is started. The initialization value is 1.
R 3	The one's complement of the number of error location polynomial coefficients less 3. It is used as a counter to determine when t-2 roots have been found.
R 4	Address of current erasure. It is initially set to either 71(Ø) or to 271(Ø) . It is incremented by one each time the updated λ substitution value equals an erasure location.
R 5	The one's complement of the number of coefficients of the error location polynomial.
R 6	The row counter. Initially set to the one's complement of 37(Ø).
R 7	The column term. See decription of row/column processing in section 4.6.1 .
R 8	The current erasure. Used to check for equality with λ substitution value.
R 9	Current data block indicator Equals 0 when A is current data block Equals 200(Ø)when B is current data block.
T	

CPE Registers as used by EPVAL during the second processing phase.

TABLE 4.12 Page 2/3

CPE REGISTERS USED AS INDEXES BY : EPVAL

REGISTER	CONTENT/PURPOSE
R 0	The upper limit of the trial pair in the two root solution. It is decremented by 1.
R 1	The lower limit o the trial pair in the two root solution. It is incremented by 1.
R 2	SIRCS. The cummulative inverse roots code sum.
R 3	SIRPS. The cummulative inverse roots power product.
R 4	Equals p defined as the coefficient of highest degree term minus the cummulative inverse roots power product of t-2 roots. i.e. σ_t - SIRPS.
R 5	The one's complement of the number of coefficients of the error location polynomial.
R 6	Code form of the coefficient of the lowest degree term of the error location polynomial.
R 7	Power form of the coefficient of the highest degree term of the error location polynomial. Is used later as the second set of limits indicator.
R 8	Current data block code select(long or short) indicator Equals 4 when A is current data block Equals 204(Ø) when B is current data block.
R 9	

CPE Registers as user by EPVAL during the two root processing.

TABLE 4.12 Page 3/3

4.6.4 Input/Output Formats

Inputs and outputs processed by EPVAL are listed below together with their common data base references which describe them in detail:

ITEM	INPUT	OUTPUT	REFERENCE
IADCC/IBDCC	X		TADIO/TBDIO
IADCS/IBDCS	X		TADIO/TBDIO
IADEE	X		TADIO
IBDEE	X		TBDIO
IADES	X		TADIO
IBDES	X		TBDIO
SCBIX	X		SCBIX
SEPEP		X	SEPEP
TEPOL + 0 to + 30	X		TEPOL
TEVAL + 0 to + 30		X	TEVAL
R_9	X	X	Table 4.12
TCTPC	X		TCTPC
TPTCC	X		TPTCC
ACCUMULATOR	X		Contains address of the error location polynomial.

4.6.5 Conditions for Initiation

Subprogram EPGEN passes control to EPVAL when it has completed its own functions. EPVAL is always entered as part of the decoding process unless EPGEN has made one of the following abnormal exits :

- o A decode failure exit to subprogram INOUT. (2e+E>16)
- o An exit is made to subprogram RPVAL when E=16.
- o An exit is made to subprogram RPVAL when e=0.

4.6.6 Limitations

EPVAL does not have any special limitations.

4.6.7 Interface Description

Reference Figure 3.3 for a block diagram showing the sequential and functional relationship of all subprograms.

Subprogram EPGEN stores the address of its output, the error location polynomial, in the CPE accumulator. It then passes control to EPVAL.

EPVAL computes the locations of the errors and then passes control to subprogram RPGEN. EPVAL makes an abnormal exit to INOUT if it has not found sufficient root solutions to the error polynomial. The entry to RPGEN is at entry point RPG00 . The entry point to INOUT is the decode failure entry point, IOC01.

4.7 Subprogram RPGEN

4.7.1 Detailed Description

The function of subprogram RPGEN is to compute the errata polynomial. The errata (i.e. the combination of errors and erasures) polynomial is a function of the syndromes, the erasure polynomial and the error polynomial. It is defined as $\tilde{Z}(X)$ where $Z(X)$ is defined as:

$$Z(X) = (1 + S(X)) \sigma_E(X) \sigma_e(X)$$

where γ_i are the coefficients of $Z(X)$

and $$\tilde{Z}(X) = X^{s+t} Z(1/X)$$

$$= \sum_{i=s+t-A}^{s+t} \gamma_{s+t-i} X^i \qquad \text{where } A \leq s+t$$

Only the coefficients $X^0, X^1, \ldots\ldots X^{s+t}$ are required.

and

$S(X)$ = The syndromes $S_1, S_2, \ldots\ldots\ldots\ldots S_{16}$

$\sigma_E(X)$ = The coefficients of the erasure polynomial

$\sigma_e(X)$ = The coefficients of the error polynomial

s = The number of erasures

Subprogram MSYNG has already perfomed the computation of

$$(1 + S(X)) \sigma_E(X)$$

and has stored the results in Table TMSYN. s is the number of erasures and is stored in word 0 of Table TMSYN. In addition, MSYNG converted the code values stored in Table TMSYN to power and stored them in Table TMSYP.

TMSYN and TMSYP are input for RPGEN which then proceeds to complete the computation by multiplying all the terms (exclude word 0) by the coefficients of the error polynomial, stored in Table TEPOL. Refer to section 4.4 for the details of the processing by MSYNG.

When control is passed to RPGEN, words 1, 2.....s+t of TMSYN contain the values, in code form , of $T_1, T_2, T_3, \ldots\ldots T_{s+t}$

where t is the number of errors. TMSYP contains the same T terms in code form.RPGEN takes terms from TMSYN for a Galois add (XOR) and from TMSYP for a Galois multiply (add).

In the first pass, the first error coefficient σ_1 is added to T_1 and stored in word 1. T_1 is then multiplied by σ_1 and added to T_2 in word 2. T_2 is multiplied by σ_1 and added to T_3 in word 3. This continues until T_{s+t-1} is multiplied by σ_1 and added to T_{s+t}. Subscripts of the terms multiplied together always sum to the T subscript to which they are added which also equals the word number in which they are stored. The outputs are written back into Table TMSYN.

On the next pass, the second error coefficient σ_2 is added to the term in word 2 which then becomes $T_2 + T_1\sigma_1 + \sigma_2$. T_1 is now multiplied by σ_2 and added to word 3. T_2 is multiplied by σ_2 and added to word 4. This continues until T_{s+t-2} is multiplied by σ_2 and added to word s+t.

The processing continues in this way, the last pass beginning with σ_t being added to word t and the product $T_1\sigma_t$ being added to word t+1.

Reference Table 4.13

RPGEN writes the total number of errata (errors plus erasures) into word zero of Table TMSYN.

If there are no errors then no processing is required by RPGEN.

When RPGEN has completed its processing it exits to RPVAL at entry point point RPV00 and gives control to RPVAL.

4.7.2 Flow Diagram

Figure 4.7 is the flow diagram for subprogram RPGEN.

4.7.3 Environment

4.7.3.1 Tables

RPGEN uses only common data base tables.

4.7.3.2 Variables

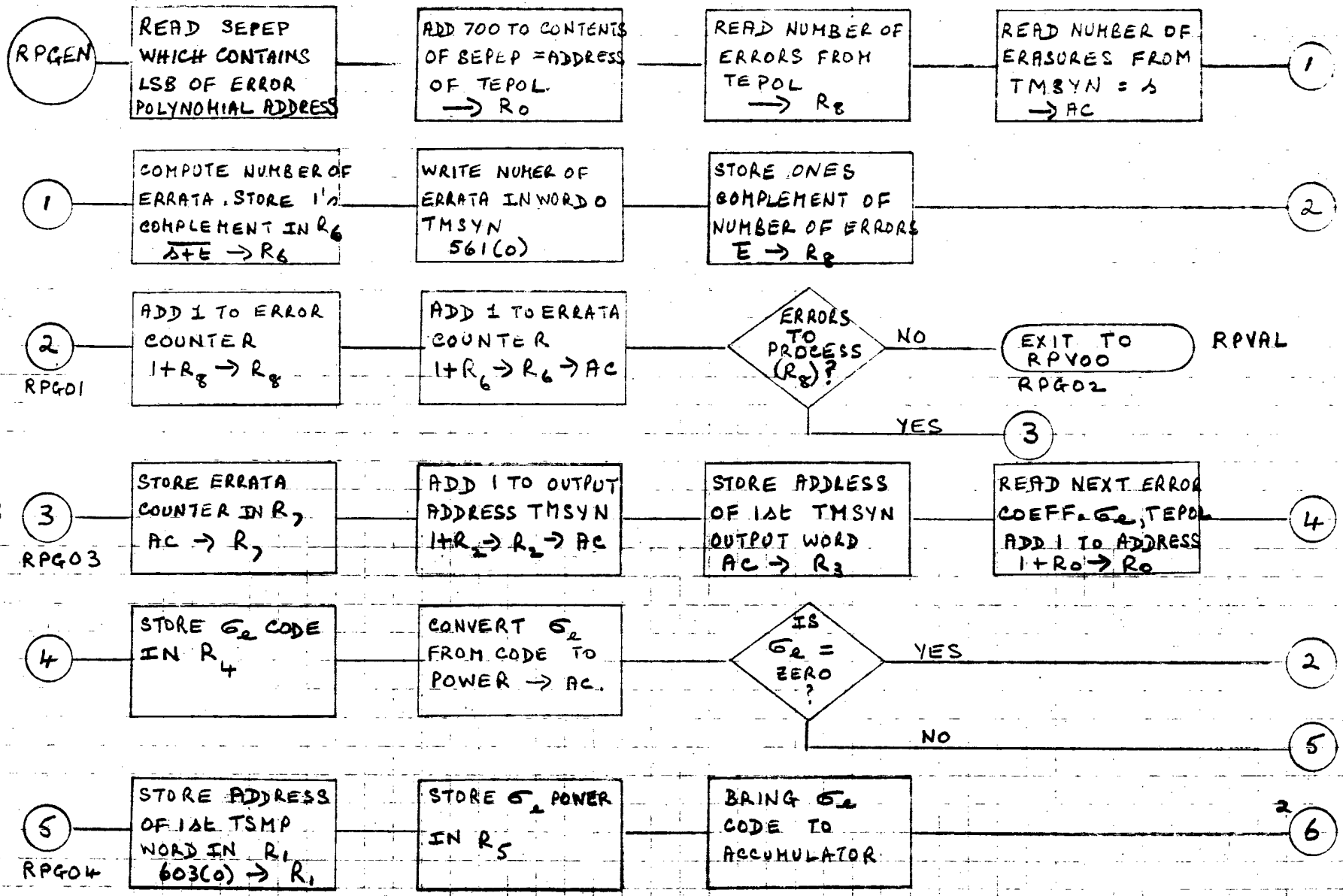
RPGEN uses only common data base variables.

ERRATA POLYNOMIAL GENERATION

TMSYN	PASS 0	PASS 1	PASS 2	.....PASS
Word 0				
Word 1	T_1	$T_1 + \sigma_1$	$T_1 + \sigma_1$	
Word 2	T_2	$T_2 + T_1\sigma_1$	$T_2 + T_1\sigma_1 + \sigma_2$	
Word 3	T_3	$T_3 + T_2\sigma_1$	$T_3 + T_2\sigma_1 + T_1\sigma_2$	
Word 4	T_4	$T_4 + T_3\sigma_1$	$T_4 + T_3\sigma_1 + T_2\sigma_2$	
Word 5	T_5	$T_5 + T_4\sigma_1$	$T_5 + T_4\sigma_1 + T_3\sigma_2$	
.	.	.	.	
.	.	.	.	
.	.	.	.	
.	.	.	.	
Word s+t	T_{s+t}	$T_{s+t} + T_{s+t-1}\sigma_1$	$T_{s+t} + T_{s+t-1}\sigma_1 + T_{s+t-2}\sigma_2$	

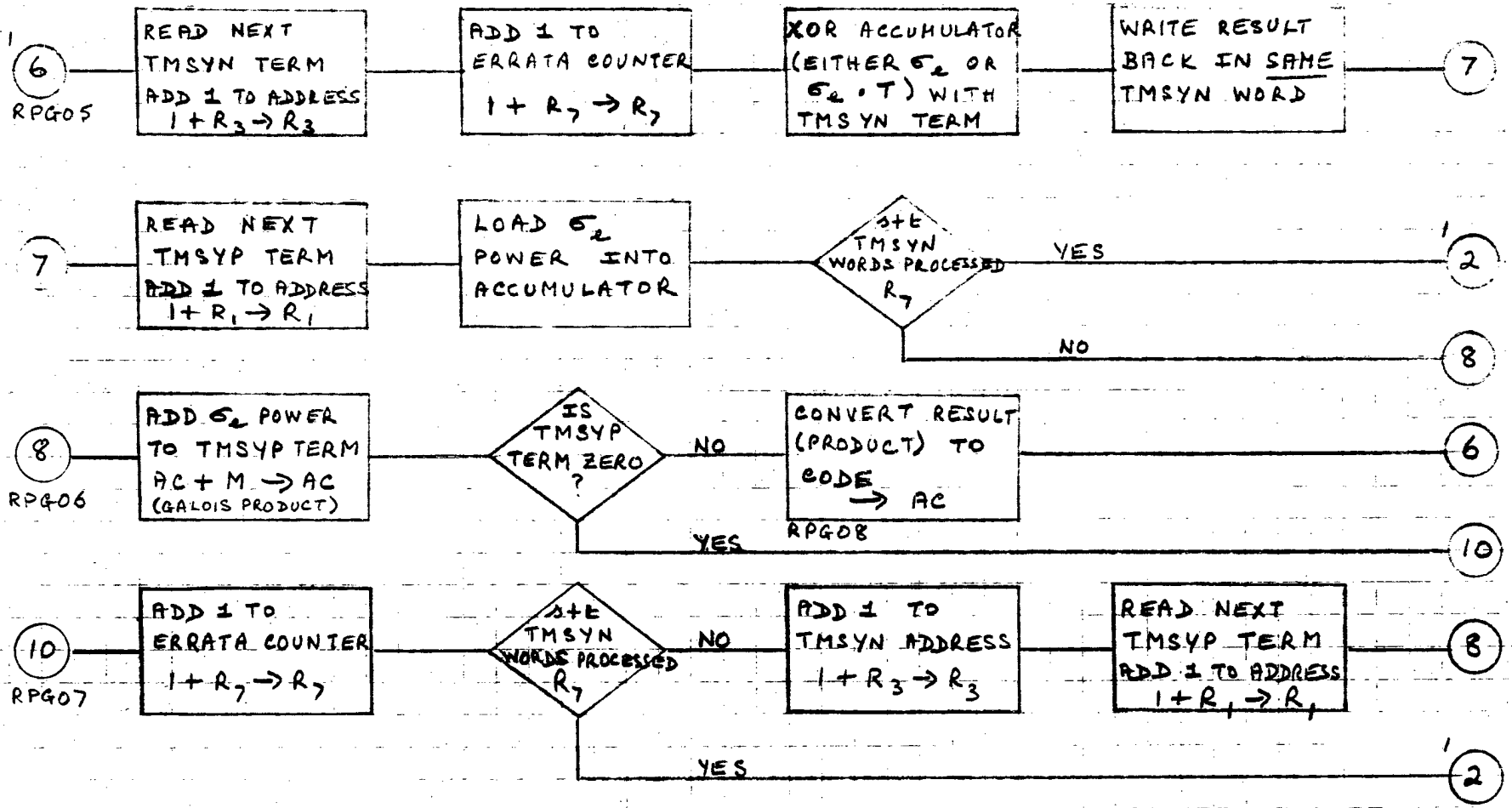
1. T = Output of MSYNG stored in TMSYN.
2. s = the number of erasures. t = the number of errors.
3. σs are the coefficients of the error polynomial stored in Table TEPOL (TAPAAor TAPBB or TAPCC)
4. Reference section 4.4 for a description of the computation of the T terms by MSYNG and in particular reference Table 4.7
5. Illustration of entries into Table TMSYN during RPGEN processing.

TABLE 4.13



SUBPROGRAM RPGEN FLOW DIAGRAM

FIGURE 4.7 Page 1/2



SUBPROGRAM RPGEN FLOW DIAGRAM

FIGURE 4.7 Page 2/2

4.7.3.3 Constants

RPGEN does not use any constants other than the use of the K bus which is defined in the appropriate instruction.

4.7.3.4 Flags

RPGEN uses only common data base flags.

4.7.3.5 Indexes

Table 4.14 describes the indexes used by RPGEN.

4.7.3.6 Common Data Base Reference

The following common data base items are referenced by RPGEN:

ITEM	TABLE
SEPEP	SEPEP
TEPOL + 0 to + 27(D)	TEPOL
TMSYN + 0 to + 16(D)	TMSYN
TMSYP + 0 to + 16(D)	TMSYP
TCTPC/TPTCC	TCTPC/TPTCC

4.7.4 Input/Output Formats

Inputs and outputs processed by RPGEN are listed below together with their common data base reference which describes them in detail:

ITEM	INPUT	OUTPUT	REFERENCE
SEPEP	X		SEPEP
TEPOL + 0 to + 27(D)	X		TEPOL
TMSYN + 0 to + 16(D)	X	X	TMSYN
TMSYP + 0 to + 16(D)	X		TMSYP
TCTPC/TPTCC	X		TCTPC/TPTCC

CPE REGISTERS USED AS INDEXES BY : RPGEN

REGISTER	CONTENT/PURPOSE
R_0	Address of error polynomial TEPOL
R_1	ADDress of TMSYP. It is incremented during each inner loop and reset to the address of the first word at the start of each pass.
R_2	Address of TMSYM.It is incremented by 1 for each pass and used to indicate the starting point in TMSYN for the next pass.
R_3	Address of TMSYM. It is used to read terms from TMSYN and then to write them back into TMSYN during each pass. It is reset from R_2 at the start of the next pass.
R_4	Current σ_e error coefficient in code form.
R_5	Current σ_e error coefficient in power form.
R_6	One's complement of the number of errata.
R_7	One's complement of the number of errata. It is incremented in the inner loop and used to indicate that all necessary T terms have been multiplied by current σ_e. It is reset from R_6 at the start of each pass.
R_8	One's complement of the number of errors. It is incremented by 1 on each pass and controls the number of passes to be performed.
R_9	
T	

TABLE 4.14

4.7.5 Conditions for Initiation

EPVAL passes control to RPGEN when it has completed its own functions. RPGEN is always entered from EPVAL as part of the decode process unless EPVAL has made an abnormal exit to INOUT in the case of failure to find sufficient roots.

4.7.6 Limitations

RPVAL does not have any special limitations

4.7.7 Interface Description

Reference Figure 3.3 for a block diagram showing the sequential and functional relationship of all subprograms.

RPGEN computes the errata polynomial using terms generated earlier by subprogram MSYNG. When it has completed its function RPGEN always exits to RPV00 passing control to RPVAL.

4.8 Subprogram RPVAL

4.8.1 Detailed Description

The function of subprogram RPVAL is to compute the errata values. An errata value is defined as the value which must be added to an erroneous received character in the R/S word being decoded. Addition in this process is Galois addition, which is equivalent to an exclusive or function. Received data characters which are determined to be either erasures or errors will be corrected. The received parity characters will not be corrected.

If β_{E_i} and β_{e_i} are the erasure and error locations, then

the errata locations are defined as

$$\beta_i \text{ , for } i = (E_i, e_i)$$

The errata correction value C_i is defined as

$$C_i = \frac{\tilde{Z}(\beta_i)}{\beta_i \prod_{i \neq j} (\beta_i + \beta_j)} \qquad \text{for all } i,j = E_i \text{ or } e_i$$

and

$$C_i = \emptyset \ (00000) \qquad \text{for all } i,j \neq E_i \text{ or } e_i$$

$$\tilde{Z}(X) = \tilde{Z}(\beta_i) = X^{s+t} Z(1/X)$$

RPVAL examines the erasures stored in the current input/output area by the associated RSED hardware and separates them into data and parity characters by virtue of their position number. The data characters are stored sequentially in power form in Table TRVOB. All characters (data plus parity) are converted to code form and stored sequentially in Table TRVOA.

RPVAL then examines the errors, input from subprogram EPVAL, stored in Table TEVAL. RPVAL selects the errors in the data characters and stores them sequentially following the erasure characters, in Table TRVOB in power form. All (data plus parity) characters are then converted to code form and stored in Table TRVOA sequentially following the erasure characters stored there.

RPVAL computes the total number of errata (errors plus erasures) in the data characters and stores the result in Table TRVOC, an intermediate output table.

For each errata stored in table TRVOB, an errata value will be calculated. The value

$$\beta_i \prod_{i \neq j} (\beta_i + \beta_j)$$

will be calculated first. The terms in Table TRVOB are represented by the i values and the terms in TRVOA are represented by the j values. Since the code value of the zero term, a non-Galois Field element,is defined for convenience in Tables TCTPC and TPTCC as zero, then the restriction in the above product of $i \neq j$ can be virtually ignored. When $i = j$, the Galois field sum of $\beta_i + \beta_j$ will equal zero. This zero value will then be Galois field multiplied , that is added, to the cummulative product. The addition of a zero term will leave the product unaltered. This is done as a time saving device removing the necessity for zero testing each time a new j term is processed. An i term is read from Table TRVOB and stored. The first j term is read from Table TRVOA and a Galois add (exclusive or) is perfomed with the stored i term. The result is then converted to power form and cummulatively Galois field multiplied (added) into the product for the corresponding i value. The product is stored in a CPE register, initialized to the power value of the corresponding i value. The second errata j term is read from Table TRVOA and processed in the same way.The product for the current i term is complete when all j terms have been Galois multiplied with the i term and included in the product.When this has occured the result is written in power form in Table TRVOC. The second i term in Table TRVOB is then stored and a product computed for it in the same way. The processing continues in this way until products have been computed for all i terms.

RPVAL now computes the numerator $\tilde{Z}(\beta_i)$. $\tilde{Z}(X)$ is the inverse of the errata polynomial $Z(X)$ computed by subprogram RPGEN and stored in Table TMSYN. Writing out the terms of the polynomials we have:

$$\tilde{Z}(X) = X^{s+t} Z(1/X)$$

$$Z(X) = 1 + \gamma_1 X + \gamma_2 X^2 + \ldots\ldots\ldots\ldots\ldots\ldots \gamma_{s+t} X^{s+t}$$

$$\tilde{Z}(X) = X^{s+t} \left(1 + \gamma_1 / X + \gamma_2 / X^2 + \ldots\ldots\ldots\ldots + \gamma_{s+t} / X^{s+t} \right)$$

$$= X^{s+t} + \gamma_1 X^{s+t-1} + \gamma_2 X^{s+t-2} + \ldots\ldots\ldots + \gamma_{s+t}$$

Two comments about the $\tilde{Z}(X)$ polynomial are noted. The first is that the coefficients stored in Table TMSYN are the coefficients of polynomial $Z(X)$ in ascending order but are the coefficients of polynomial $\tilde{Z}(X)$ in descending order.The second comment is to note that for $\tilde{Z}(X)$ the highest order coefficient of the term X^{s+t} always has the value one. Also note that the coefficient with value one is not stored in Table TMSYN, this coefficient being the coefficient of X^0 (always equal to one) in the $Z(X)$ polynomial. RPVAL makes use of both of these facts in the computation of $\tilde{Z}(X)$.

The polynomial $\tilde{Z}(X)$ can be written as

$$\tilde{Z}(X) = ((((X + \gamma_1) X + \gamma_2) X + \gamma_3) X + \ldots\ldots \gamma_{s+t-1}) X + \gamma_{s+t}$$

and this form is used for the computation which is described below. $\tilde{Z}(\beta_i)$ is computed by consecutive substitution of the β_is.

The first errata term β_1 is read from TRVOB, stored in the CPE register R_3 in power form and is converted to code and stored in code form in the CPE T register. The first errata polynomial coefficient γ_1 is read, in code form, from Table TMSYN. It is Galois added (XOR) with the sum in the T register. The result $\beta_1 + \gamma_1$, is tested for zero value and if it equals zero, a zero is stored as the cummulative sum in the T register. If the result is not zero, it is converted to power form and then Galois multiplied (added) to β_1 stored in R_3. The result $(\beta_1 + \gamma_1)\,\beta_1$ is converted back to code form and stored as the cummulative sum in T. The second coefficient γ_2 is read in code form from Table TMSYN. It is XORed with the sum in T giving the term $(\beta_1 + \gamma_1)\,\beta_1 + \gamma_2$.

If this term is not equal to zero, then it is Galois multiplied(added) to β_1 in R_3 giving the term $((\beta_1 + \gamma_1)\,\beta_1 + \gamma_2)\,\beta_1$ which is converted to code and stored back in T. The process continues until (s+t-1) coefficients from TMSYN have been processed. The sum in T is now

$$(((\beta_1 + \gamma_1)\,\beta_1 + \gamma_2)\,\beta_1 + \ldots\ldots\ldots \gamma_{s+t-1})\,\beta_1 .$$

The final coefficient γ_{s+t} is then XORed with the sum in T. This is now the value of $\tilde{Z}(X)$ for the first errata β_1.

Since $\tilde{Z}(X)$ is a numerator it will be tested for zero value and if it is zero,the correction value for this term is zero and no further processing is performed on this character.

RPVAL now makes the computation $Z(\beta_i) / \beta_i \prod_{i \neq j} (\beta_i + \beta_j)$ for the β_i value stored in R_3. The numerator is the value stored in the CPE T register. The denominator terms for all i values are stored sequentially in TRVOC in power form. $\tilde{Z}(\beta_i)$ in the T register is first converted to power form. The denominator product term corresponding to the i term in R_3, is read from Table TRVOC and Galois field divided (subtracted) from $\tilde{Z}(\beta_i)$. If the result is negative, 31(D) is added to the result according to the rules of Galois field arithmetic. This result is converted to code form and XORed with the corresponding data character, given by the position i, in the input/output area, the corrected character being written back into this area. Processing for this character is now completed.

RPVAL then reads the next errata from Table TRVOB and proceeds to compute the errata correction value in the same way, computing first the denominator product, the numerator polynomial and then the division. As each value is computed, it is applied to the corresponding data character requiring correction.

When correction values have been computed and the adjustment applied for all β_is stored in Table TRVOB, RPVAL has completed its major function.

During this processing, RPVAL determines the point in time at which two (2) errata correction values still must be calculated. When this point is reached, RPVAL sets the WRITE control flag for the non-current input/output data area to a WRITE REQUEST. This is done to permit the transfer into the Data Ram of the next R/S code word to begin before the termination of processing on the current R/S code word. Reference section 3.1.2 of the document reference number 1.

Additionally RPVAL has a special entry point RPV02, entered by sub-program INOUT, which sets the WRITE REQUEST flag in the non-current data block. This is to handle those R/S code words which have not passed through the normal processing path e. g. decode failures, code words without errata.

RPVAL's final function is to write the data quality summary into the data quality word in the current input/output data block. Bit 4 of the word is set to the value zero to indicate that the R/S code word was sucessfully decoded. The number of errors (erasures are not included) in the data and parity characters is written into bits 3 through 0 in the same word.

After RPVAL has completed these functions, it exits to INOUT at entry point INC03 passing control to INOUT. Refer to section 4.2.1.

4.8.2 Flow Diagram

Figure 4.8 is the flow diagram for subprogram RPVAL.

4.8.3 Environment

4.8.3.1 Tables

The following tables are used only by RPVAL:

1. Table name is TRVOA

   o Purpose

     TRVOA is used by RPVAL to store the code values of all erasures and errors. It is an intermediate output table. RPVAL uses this data later in its own computation.

   o Size and Indexing Procedure

     TRVOA is of fixed length 17(D) words. It is addressed by using the K bus instruction setting to be the Data Ram address of word 0 i.e. 151(O).
     A CPE index register is set to this value and is automatically incremented by one whenever the table is written into. This provides the address of the next sequential word in the table.

   o Structure and Bit Layout.

     TRVOA is a vertical type table.
     The table format is shown in Table 4.15

2. Table name is TRVOB

   o Purpose

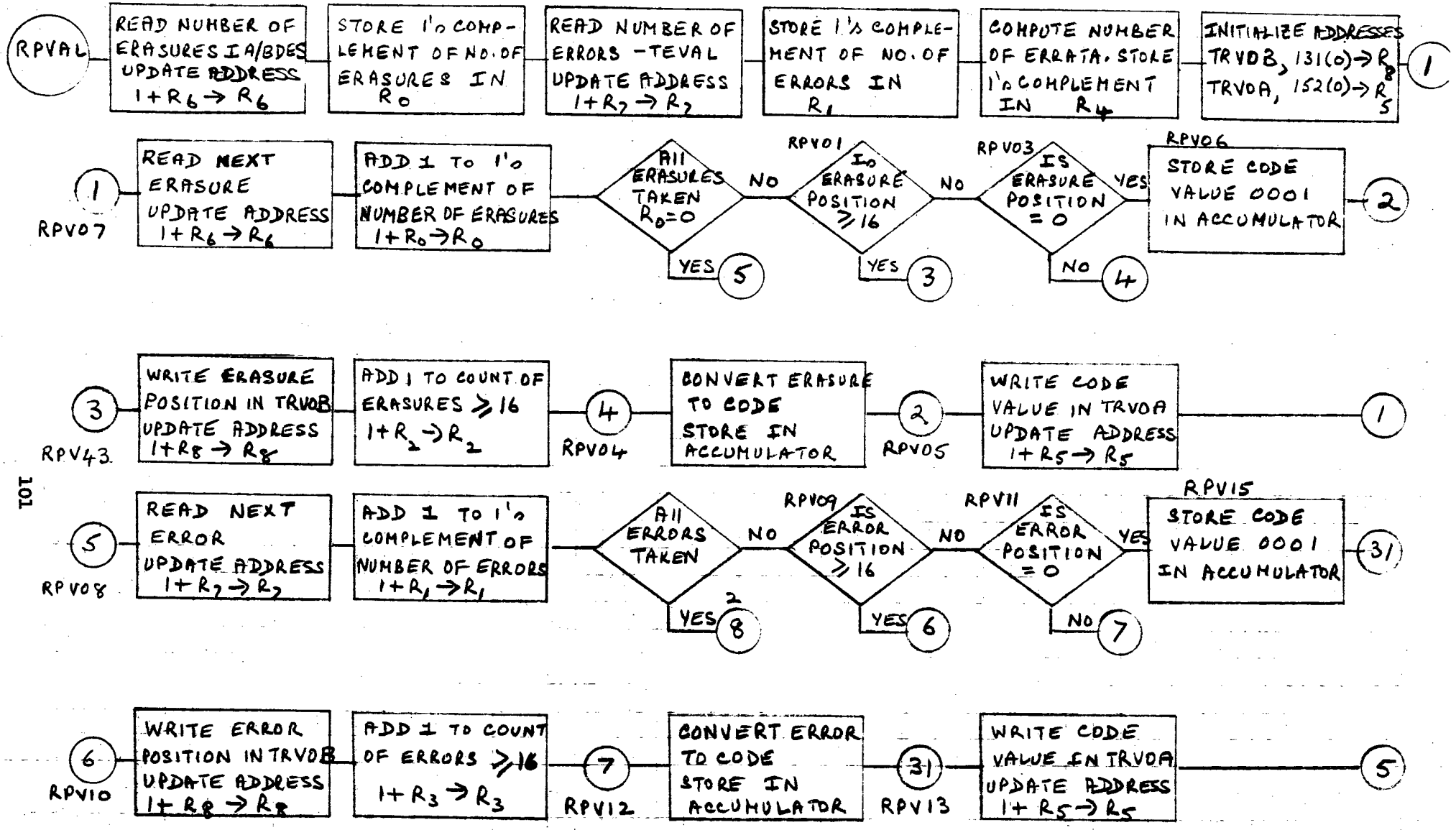
     TRVOB is used by RPVAL to store the power form of erasures and errors located in the data characters only. It is an intermediate output table. RPVAL uses this data later in its own computation.

   o Size and Indexing Procedure

     TRVOB is of fixed length 17(D) words. It is addressed in the same manner as TRVOA, described above.

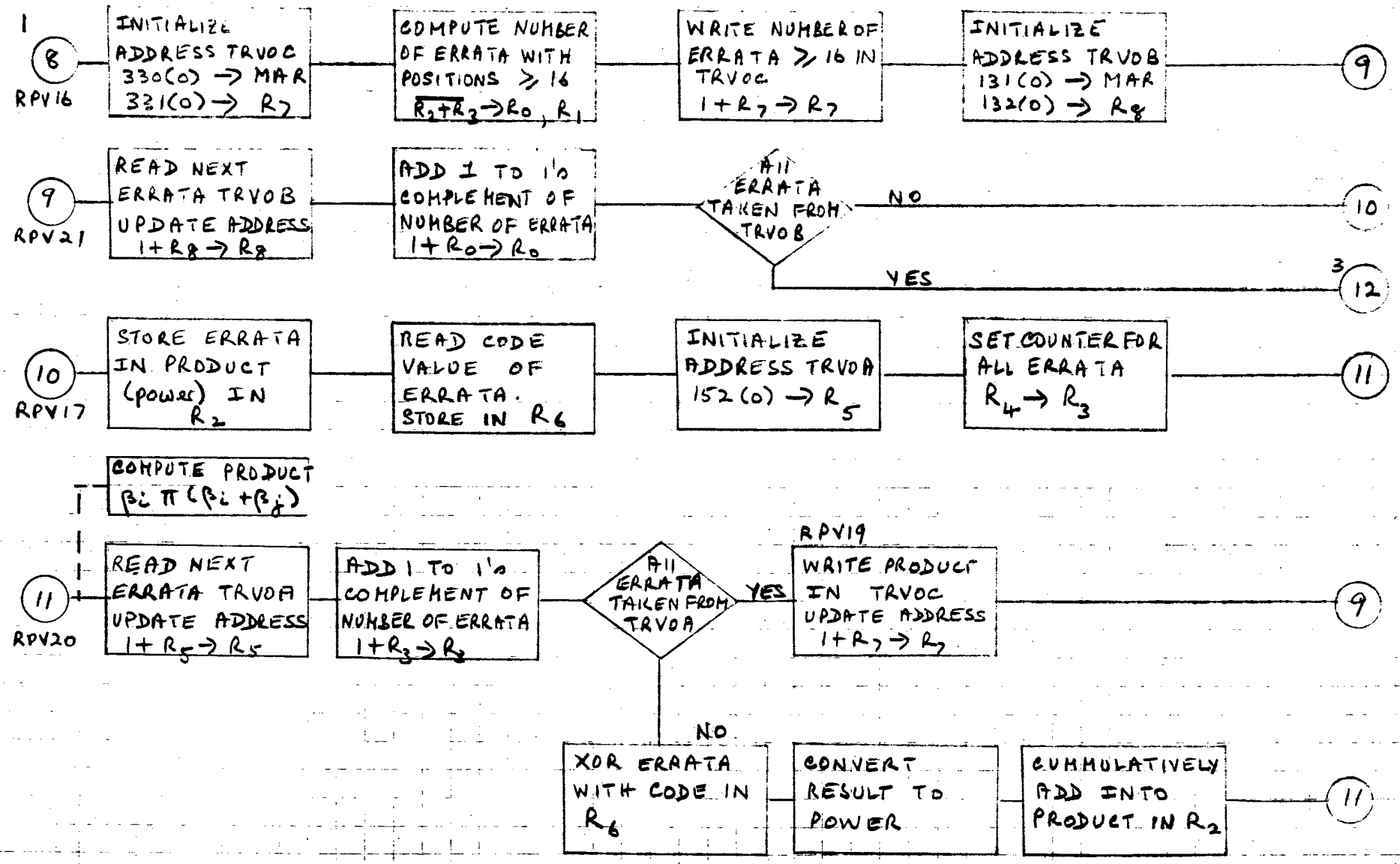
   o Structure and Bit Layout

     TRVOB is a vertical type table.
     The table format is shown in Table 4.16



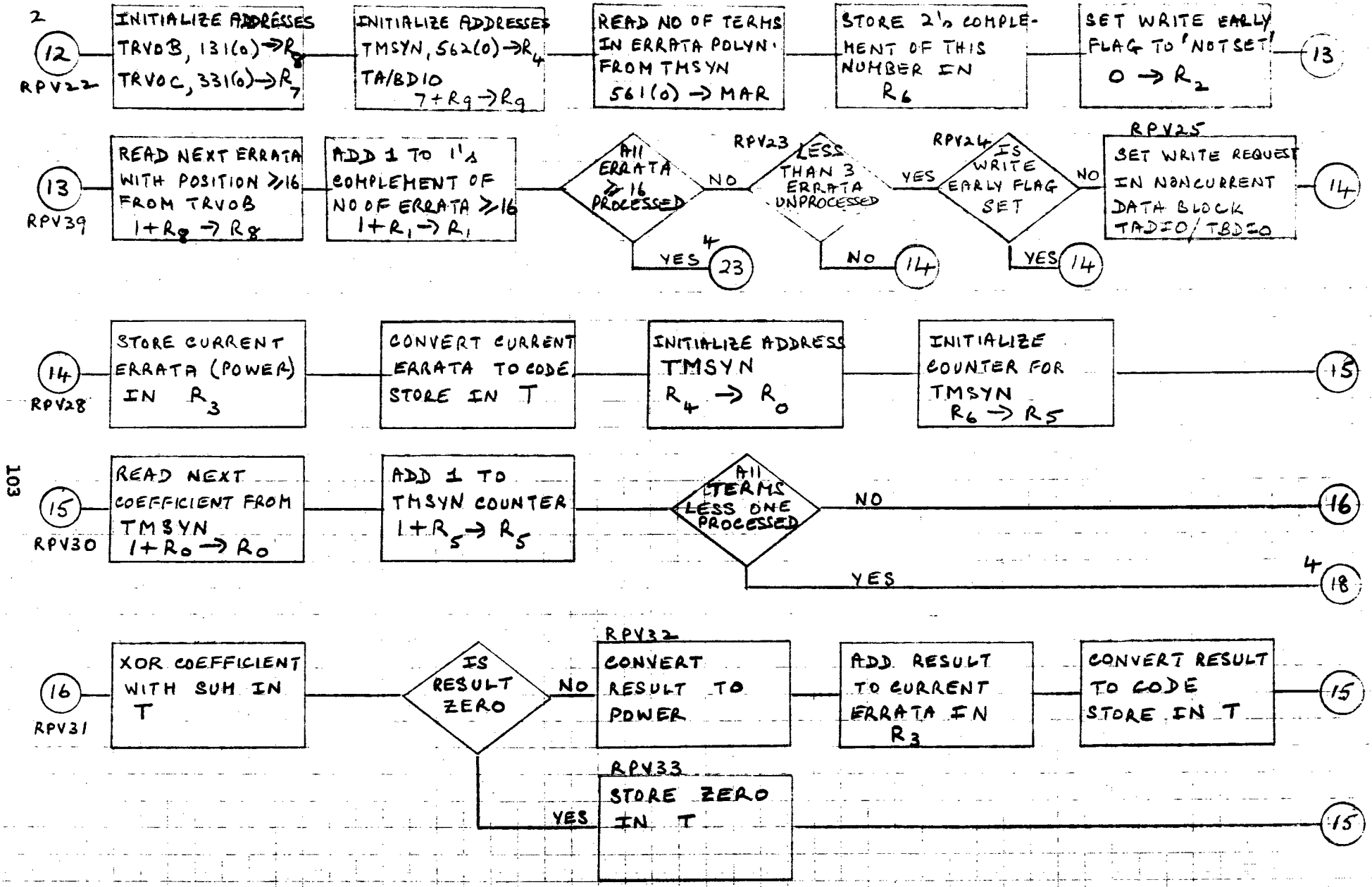
SUBPROGRAM RPVAL FLOW DIAGRAM MERGE ERRATA

FIGURE 4.8 Page 1/4



SUBPROGRAM RPVAL FLOW DIAGRAM — COMPUTE PRODUCT

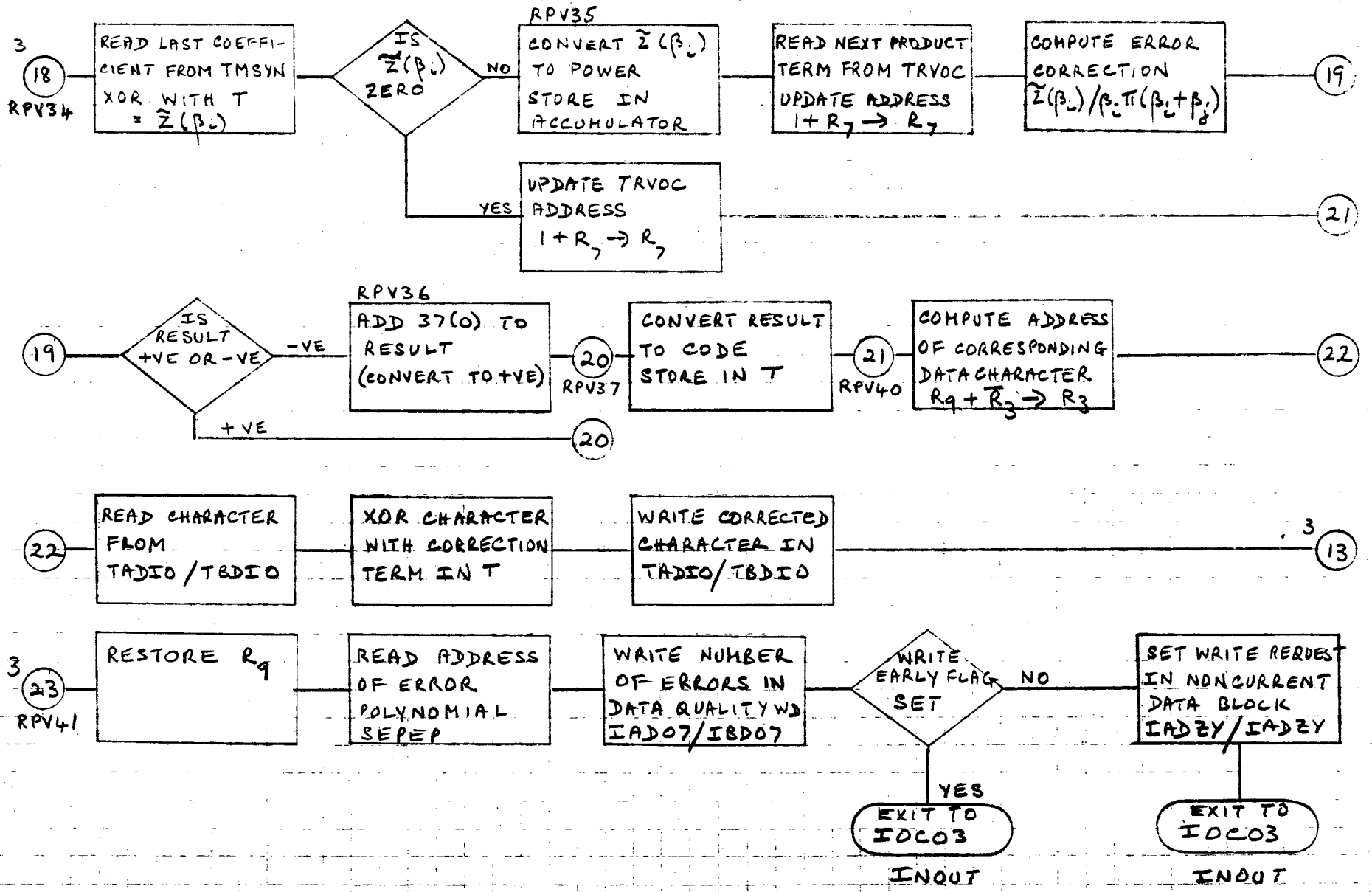
FIGURE 4.8 Page 2/ 4



103

SUBPROGRAM RPVAL FLOW DIAGRAM COMPUTE $\tilde{Z}$

FIGURE 4.8 Page 3/4



SUBPROGRAM RPVAL FLOW DIAGRAM COMPUTE CORRECTION VALUE

FIGURE 4.8 Page 4/4

TABLE TYPE: INTERNAL TABLE NAME: TRVOA

RAM ADDRESS	WORD NUMBER	BIT LOCATIONS 7 6 5 4 3 2 1 0	CONTENTS
OCTAL			
151	00	X X X	Not used
152	01	X X X	Code form of errata
153	02	X X X	Code form of errata
154	03	X X X	Code form of errata
155	04	X X X	Code form of errata
156	05	X X X	Code form of errata
157	06	X X X	Code form of errata
160	07	X X X	Code form of errata
161	08	X X X	Code form of errata
162	09	X X X	Code form of errata
163	10	X X X	Code form of errata
164	11	X X X	Code form of errata
165	12	X X X	Code form of errata
166	13	X X X	Code form of errata
167	14	X X X	Code form of errata
170	15	X X X	Code form of errata
171	16	X X X	Code form of errata

X = not used

TABLE 4.15

TABLE TYPE: INTERNAL TABLE NAME: TRVOB

RAM ADDRESS OCTAL	WORD NUMBER	BIT LOCATIONS 7 6 5 4 3 2 1 0	CONTENTS
130	00	X X X	Not used.
131	01	X X X	Power form of data character errata.
132	02	X X X	•
133	03	X X X	•
134	04	X X X	•
135	05	X X X	•
136	06	X X X	•
137	07	X X X	•
140	08	X X X	•
141	09	X X X	•
142	10	X X X	•
143	11	X X X	•
144	12	X X X	•
145	13	X X X	•
146	14	X X X	•
147	15	X X X	•
150	16	X X X	Power form of data character errata.

X = not used

TABLE 4. 16

3. Table name is TRVOC

- o Purpose

  TRVOC is used by RPVAL to store the product terms β_i II $(\beta_i + \beta_j)$ in power form. It is an intermediate output table. RPVAL uses this data later in its own computations.

- o Size and Indexing Procedure

  TRVOC is of fixed length 17(D) words. It is addressed in the same manner as TRVOA, described above.

- o Structure and Bit Layout

  TRVOC is a vertical type table.
  The table format is shown in Table 4.17

4.8.3.2 Variables

RPVAL uses only common data base variables

4.8.3.3 Constants

RPVAL does not use any constants other than the use of the K bus which is defined in the appropriate instruction.

4.8.3.4 Flags

RPVAL uses CPE register R_2 as a flag. See Table 4.18 for a description.

4.8.3.5 Indexes

Table 4.18 describes the indexes used by RPVAL.

4.8.3.6 Common Data Base Reference

The following data base items are referenced by RPVAL:

ITEM	TABLE
IAD07	TADIO
IBD07	TBDIO
IADCS + 0 to + 30	TADIO
IBDCS + 0 to + 30	TBDIO
IADEE + 0 to + 30	TADIO
IBDEE + 0 to + 30	TBDIO

TABLE TYPE: I INTERNAL TABLE NAME: TRVOC

RAM ADDRESS OCTAL	WORD NUMBER	BIT LOCATIONS 7 6 5 4 3 2 1 0	CONTENTS
330	00	X X X	Number of errata in data characters.
331	01	X X X	Product term β_i II ($\beta_i + \beta_j$) in power form.
332	02	X X X	•
333	03	X X X	•
334	04	X X X	•
335	05	X X X	•
336	06	X X X	•
337	07	X X X	•
340	08	X X X	•
341	09	X X X	•
342	10	X X X	•
343	11	X X X	•
344	12	X X X	•
345	13	X X X	•
346	14	X X X	•
347	15	X X X	•
350	16	X X X	Product term β_i II ($\beta_i + \beta_j$) in power form.

X = not used

TABLE 4.17

CPE REGISTERS USED AS INDEXES BY : RPVAL

REGISTER	CONTENT/PURPOSE
R 0	One's complement of the number of erasures. Used as a counter.
R 1	One's complement of the number of errors. Used as a counter.
R 2	The number of erasures in the data characters.
R 3	The number of errors in the data characters.
R 4	One's complement of the number of errata i.e. errors plus erasures.
R 5	Address of errata table TRVOA. Initially set to 152(O). Incremented by one each time an errata is read.
R 6	Address of erasures in table TADIO/TBDIO. Initially set to 70(O) or 270(O). Incremented by one each time an erasure is read.
R 7	Address of error table TEVAL. Initially set to 646(O). Incremented by one each time an error is read.
R 8	Address for errata in data characters table TRVOB. Initially set to 131(O). Incremented by one each time an errata is read.
R 9	Current data block indicator. Equals 0 when A is current data block. Equals 200(O) when B is current data block.
T	

CPE Registers used by RPVAL during the selection of errata in data characters processing.

TABLE 4.18 Page 1/3

CPE REGISTERS USED AS INDEXES BY : RPVAL

REGISTER	CONTENT/PURPOSE
R 0	One's complement of number of errata in data characters. Used as a counter for table TRVOB.
R 1	One's complement of the number of errata in data characters. Used to reset the counter R_0.
R 2	The cummulative product for current β_i errata term. β_i II ($\beta_i + \beta_j$)
R 3	One's complement of the total number of errata. Used as a counter for table TRVOA.
R 4	One's complement of the total number of errata. Used to reset counter R_3.
R 5	Address of errata table TRVOA. Initially set to 152(O). Incremented by one each time table is addressed.
R 6	Code value of current errata.
R 7	Address of product table TRVOC. Initially set to 330(O). Incremented by one each time table is addressed.
R 8	Address of table TRVOB. Initially set to 131(O). Incremented by one each time table is addressed.
R 9	Current data block indicator. Equals 0 when A is current data block. Equals 200(O) when B is current data block.
T	

CPE Registers used by RPVAL during computation of Product β_i II ($\beta_i + \beta_j$)

TABLE 4. 18 Page 2/3

CPE REGISTERS USED AS INDEXES BY : RPVAL

REGISTER	CONTENT/PURPOSE
R 0	Address for table TMSYN. Initially set to 562(O). Used to read terms from TMSYN. Incremented by one for each read.
R 1	One's complement of number of errata in the data characters. Used as a counter (incremented by one) to terminate the processing.
R 2	Write early flag indicator for current word. Equals 0 when Write Request not made. Equals not 0 when Write Request is made.
R 3	Power form of current errata β_i.
R 4	Address of errata polynomial table TMSYN.Initially set to 562(O). Used to reset R_0.
R 5	One's complement of the number of coefficients in errata polynomial. Used as a counter for table TMSYN.
R 6	One's complement of number of coefficients in errata polynomial . Used to reset R_5.
R 7	Address of output table TRVOC. Initially set to 331(O). Incremented by one each time an item is read.
R 8	Address for errata table TRVOB. Initially set to 131(O). Incremented by one each time a term is read.
R 9	Address of data quality word in current data block. Equals 7 when A is current data block. Equals 207(O)when B is current data block.
T	

CPE Registers used by RPVAL during computation of $Z(\beta_i) / \beta_i \text{ II } (\beta_i + \beta_j)$ and correction of data characters.

TABLE 4.18 Page 3/3

ITEM	TABLE
IADES	TADIO
IBDES	TBDIO
SEPEP	SEPEP
TCTPC	TCTPC
TPTCC	TPTCC
TEVAL+ 0 to + 8	TEVAL
TMSYN+ 0 to + 16	TMSYN
IADZY	TADIO
IBDZY	TBDIO

4.8.4 Input/Output Formats

Inputs and outputs processed by RPVAL are listed below together with their common data base references which describe them in detail:

ITEM	INPUT	OUTPUT	REFERENCE
IAD07		X	TADIO
IBD07		X	TBDIO
IADCS+0 to + 30	X	X	TADIO
IBDCS+0 to + 30	X	X	TBDIO
IADEE+0 to + 30	X		TADIO
IBDEE+0 to + 30	X		TBDIO
IADES	X		TADIO
IBDES	X		TBDIO
SEPEP	X		SEPEP
TEVAL+0 to + 8	X		TEVAL
TMSYN+0 to + 16	X		TMSYN

ITEM	INPUT	OUTPUT	REFERENCE
TCTPC	X		TCTPC
TPTCC	X		TPTCC
R_9	X	X	Table 4.18
IADZY/IBDZY		X	TADIO/TBDIO

4.8.5 Conditions for Initiation

Subprogram RPGEN passes control to RPVAL after completing its own functions. RPVAL is always entered as part of the decoding process unless one of the subprograms which are executed prior to RPGEN exits to INOUT on a decode failure or a word without errata.

4.8.6 Limitations

RPVAL does not have any special limitations.

4.8.7 Interface description

Reference Figure 3.3 for a block diagram showing the sequential and functional relationship of all subprograms.

The normal entry into RPVAL is from subprogram RPGEN at entry point RPV00 which occurs whenever control is passed to subprogram RPGEN. In addition , subprogram EPGEN makes a direct entry into RPVAL if either of the two following conditions are detected:

- o There are 16 erasures. E = 16
- o There are no errors. e = 0

In both cases, the entry point is RPV00.

Subprogram INOUT also makes an entry into RPVAL but only under the condition that control was returned to INOUT at entry point IOC01 i.e. when the normal processing path was not followed. INOUT enters RPVAL at entry point RPV02 in order to get the WRITE REQUEST flag set. Refer to section 4.2.1.

Subprogram RPVAL always exits to INOUT at entry point IOC03 after completing its functions.

5.0 COMMON DATA BASE

5.1 Tables

5.1.1 Input/Output Data Block (TADIO/TBDIO)

- o Table name is TADIO/TBDIO

- o Purpose

  Two areas are reserved in the data Ram into which the associated RSED hardware deposits the R/S code word to be decoded. TADIO and TBDIO are these two areas.Reference document number 1 , Data Item A004, sections 3.1.2 and 3.1.3

- o Size and Indexing

  The tables are each of fixed length occupying fixed locations in the data Ram. Each table is 87(D) words long. Reference Table 5.1. The tables are addressed by using the K bus instruction setting to be the data Ram address of the first word of any group of items contained within the table, the tables being of vertical type. A CPE index register is set to this value and is automatically incremented by one whenever the table is addressed . This provides the address of the next sequential word.

- o Fields

  Table 5.1 describes the fields in detail. All fields may change in value except where indicated. Decimal points and scaling are not involved.

5.1.2 Data Read/Write Control (TADRW/TBDRW)

- o Table name is TADRW/TBDRW

  Purpose

  The tables contain the control flags used for reading into and out of the data Ram.Reference document number 1 .

- o Size and Indexing

  Each table is two words long and occupies fixed locations in the data Ram. Refer to Table 5.2
  The tables are addressed by using the absolute address of the data Ram which is fixed due to the associated RSED hardware requirements.

- o Fields

  Table 5.2 gives a complete description of the flags.
  Decimal points and scaling are not applicable.

INPUT/OUTPUT DATA BLOCK (TADIO/TBDIO)

[redacted] ADDRESS	BIT POSITION *5 4 3 2 1 0	DESCRIPTION	
0 / 200	ITEM NAME		
1			
2			
3			
4	IADCC/ IBDCC — 0 Long, 1 Short	Long or Short R/S code word (Bit 1)	
5			
6			
7	IAD07/ IBD07 — 1= Failure, 0= Decoded ← Number of errors →	Data Quality Word	
8	IADCS/ IBDCS ← MSB — Characters — LSB →	31/15 C30	16/4 "0"s
9	MSB LSB	C29	"0"s
10	MSB LSB	C28	"0"s
[redacted]	MSB LSB	C27	"0"s
12	MSB LSB	C26	"0"s
13	MSB LSB	C25	"0"s
14	MSB LSB	C24	"0"s
15	MSB LSB	C23	"0"s
16	MSB LSB	C22	"0"s
17	MSB LSB	C21	"0"s
18	MSB LSB	C20	"0"s
19	MSB LSB	C19	DATA
20	MSB LSB	C18	
21	MSB LSB	C17	
[redacted]	MSB LSB	C16	
23	MSB LSB	C15	Parity

* See last chart page.

RAM ADDRESS	BIT POSITION *5	4	3	2	1	0	DESCRIPTION	
24	ITEM NAME	MSB				LSB	31/15 C14	16/4
25		MSB				LSB	C13	
26		MSB				LSB	C12	
27		MSB				LSB	C11	
28		MSB				LSB	C10	
29		MSB				LSB	C09	PARITY
30		MSB				LSB	C08	
31		MSB				LSB	C07	
32		MSB				LSB	C06	
[illegible]		MSB				LSB	C05	
34		MSB				LSB	C04	
35		MSB				LSB	C03	
36		MSB				LSB	C02	4 Erasures
37		MSB				LSB	C01	
38		MSB				LSB	C00	
39	SPARE WORD	MSB				LSB	SPARE	
40	IADSN/ IBDSN	MSB				LSB	S1	SYNDROMES
41		MSB				LSB	S2	
42		MSB				LSB	S3	
43		MSB				LSB	S4	
[illegible]		MSB				LSB	S5	
45		MSB				LSB	S6	

*See last chart page.

RAM ADDRESS	BIT POSITION *5	4	3	2	1	0	DESCRIPTION
46	ITEM NAME	MSB				LSB	S7
47		MSB				LSB	S8
48							S9
49							S10 Syndrome
50							S11
51							S12
52							S13
53							S14
54							S15
55							S16
56	IADES/ IBDES	MSB				LSB	Number of Erasures
57	IADEE/ IBDEE						31/15 16/4
58							ERASURES
59							1. There are as many E_L's as there are erasures.
60							2. The E_L's are packed
61							
62							3. The E_L's are in descending order 30 thro 0
63							4. The maximum number of E_L's is 31.
64							
65							5. Positions are given as binary numbers
66							E10
67							E11
68							E12

*See last chart page.

RAM ADDRESS	BIT POSITION						DESCRIPTION
	*5	4	3	2	1	0	
69	ITEM NAME						E13
70							E14
71							E15
72							E16
73							E17
74							E18
75							E19
76							E20
77							E21
78							E22
79							E23
80							E24
81							E25
82							E26
83							E27
84							E28
85							E29
86							E30
87							E31

* Bit 5 is used as an Erasure Indicator which is valid only for words 8 thro 38.

DATA READ/WRITE CONTROL (TADRW/TBDRW)

FLAG	MEANING	SET BY:
X = 0	Data not ready	DSCU (on read completion)
X = 1	Data ready	Microprocessor Program (on decode completion)
ZY = 00	No write request made	Microprocessor Program (when processing begins)
ZY = 01	Write request made (DSCU may write)	Microprocessor Program (when processing of a word is near completion)
ZY = 11	DSCU has filled input buffer	DSCU (on write completion)
ZY = 10	Not used	

NOTES

1. A set of Read/Write control flags exist for each input/output area in a microprocessor

2. The flags are stored in fixed locations in the Data Ram used by each microprocessor as follows:

RAM ADDRESS (decimal)		BIT POSITION		MNEMONIC
126	X Flag	0	Area A	IADXX
127	ZY Flag	1, 0	Area A	IBDZY
254	X Flag	0	Area B	IADXX
255	ZY Flag	1, 0	Area B	IBDZY

TABLE 5.2

5.1.3 Coefficients of Erasure Polynomial (TAPOL)

- o Table name is TAPOL

- o Purpose

  Subprogram APGEN stores its output of the erasure polynomial generation in power form in table TAPOL where it is available for the next subprogram MSYNG.

- o Size and Indexing

  TAPOL is 17(D) words long (fixed length, vertical type). It is addressed by using the K bus instruction setting to be the data Ram address of the first word of the table. A CPE index register is set to this value and is automatically incremented by one whenever the table is addressed providing the address of the next sequential word.

- o Fields

  Table 5.3 describes the fields in detail. Decimal points and scaling are not involved.

5.1.4 Code to Power conversion Table (TCTPC)

- o Table name is TCTPC

- o Purpose

  Provides the means whereby the conversion from code to power form of Galois field elements is made. Constant values are stored in the table. The values are the power forms of the Galois field elements sequenced by code value.

- o Size and Indexing

  The table is 31(D) words long (fixed length, vertical type). The table is addressed by using the code value for which the corresponding power value is required as an index. The code value is added to the base address of 500(O) and the resulting address gives the correct power conversion.

- o Fields

  The fields are fixed five bit constants. Reference Table 5.4.

- o Initialization

  Subprogram INITZ copies the constant data from storage in ROM each time the RSED hardware is initiated.

TABLE NAME: TAPOL

RAM ADDRESS	WORD NUMBER	BIT LOCATIONS 7 6 5 4 3 2 1 0	CONTENTS
540	00	X X X	Number of erasures.
541	01	X X X	Coefficient of erasure polynomial in power form.
542	02	X X X	•
543	03	X X X	•
544	04	X X X	•
545	05	X X X	•
546	06	X X X	•
547	07	X X X	•
550	08	X X X	•
551	09	X X X	•
552	10	X X X	•
553	11	X X X	•
554	12	X X X	•
555	13	X X X	•
556	14	X X X	•
557	15	X X X	•
560	16	X X X	Coefficient of erasure polynomial in power form.

X = not used

TABLE 5.3

CODE TO POWER CONVERSION TABLE (TCTPC)

CCTPC

TCTPC

ROM ADDRESS (D)				CONTENT		RAM ADDRESS (O)	
R	C	P	BITS	BINARY	OCTAL	WORD	BITS
00	12	00	6-2	00000	00	500	4-0
01	12	00	6-2	11111	37	501	4-0
02	12	00	6-2	00001	01	502	4-0
03	12	00	6-2	10010	22	503	4-0
04	12	00	6-2	00010	02	504	4-0
05	12	00	6-2	00101	05	505	4-0
06	12	00	6-2	10011	23	506	4-0
07	12	00	6-2	01011	13	507	4-0
08	12	00	6-2	00011	03	510	4-0
09	12	00	6-2	11101	35	511	4-0
10	12	00	6-2	00110	06	512	4-0
11	12	00	6-2	11011	33	513	4-0
12	12	00	6-2	10100	24	514	4-0
13	12	00	6-2	01000	10	515	4-0
14	12	00	6-2	01100	14	516	4-0
15	12	00	6-2	10111	27	517	4-0
00	13	00	6-2	00100	04	520	4-0
01	13	00	6-2	01010	12	521	4-0
02	13	00	6-2	11110	36	522	4-0
03	13	00	6-2	10001	21	523	4-0
04	13	00	6-2	00111	07	524	4-0
05	13	00	6-2	10110	26	525	4-0
06	13	00	6-2	11100	34	526	4-0
07	13	00	6-2	11010	32	527	4-0
08	13	00	6-2	10101	25	530	4-0
09	13	00	6-2	11001	31	531	4-0
10	13	00	6-2	01001	11	532	4-0
11	13	00	6-2	10000	20	533	4-0
12	13	00	6-2	01101	15	534	4-0
13	13	00	6-2	01110	16	535	4-0
14	13	00	6-2	11000	30	536	4-0
15	13	00	6-2	01111	17	537	4-0

TABLE 5.4

5.1.5 Coefficients of Error Polynomial ((TEPOL: (TAPAA, TAPBB, TAPCC))

- Table name is TEPOL. This is the generic name for either of three tables TAPAA, TAPBB, TAPCC.

- Purpose

  EPGEN computes the coefficients of the error polynomial and its final output is in one of three areas, TAPAA or TAPBB or TAPCC. In order to save time , the data is not shifted but the address of TEPOL is varied . Subprogram EPVAL takes its input from TEPOL.

- Size and Indexing

  The total size of the three tables is 27(D) words, each table being 9 words long. The tables are fixed length , vertical type. Addressing is done by using the K bus instruction setting to be the data Ram address of the first word of either TAPAA or TAPBB or TAPCC. A CPE index register is set to this value and is automatically incremented by one whenever the table is addressed. This provides the address of the next sequential word.

- Fields

  Table 5.5 describes the fields in detail.
  Decimal points and scaling are not involved.

5.1.6 Error Locations (TEVAL)

- Table name is TEVAL

- Purpose

  EPVAL computes the locations of the errors and stores them in table TEVAL in power form to be available for RPGEN.

- Size and Indexing

  TEVAL is 9 words long (fixed length, vertical type).
  It is addressed by using the K bus instruction setting to be the data Ram address of the first word of the table. A CPE index register is set to this value and is automatically incremented by one whenever the table is addressed. This provides the address of the next sequential word.

- Fields

  Table 5.6 describes the fields in detail.
  Decimal points and scaling are not involved.

TABLE NAME: ((TEPOL (TEPAA, TEPBB, TEPCC))

RAM ADDRESS	WORD NUMBER	BIT LOCATIONS 7 6 5 4 3 2 1 0	CONTENTS
700	00	X X X	The number of coefficients of the error polynomial.
701	01	X X X	Error polynomial coefficient in code form.
702	02	X X X	•
703	03	X X X	•
704	04	X X X	•
705	05	X X X	•
706	06	X X X	•
707	07	X X X	•
710	08	X X X	Error polynomial coefficient in code form.

TEPAA is located at RAM ADDRESS 700 - 710

TEPBB is located at RAM ADDRESS 711 - 721

TEPCC is located at RAM ADDRESS 722 - 732

TABLE 5.5

TABLE NAME: TEVAL

RAM ADDRESS	WORD NUMBER	BIT LOCATIONS 7 6 5 4 3 2 1 0	CONTENTS
646	00	X X X	Number of roots found in the error polynomial.
647	01	X X X	Error location in power form.
650	02	X X X	•
651	03	X X X	•
652	04	X X X	•
653	05	X X X	•
654	06	X X X	•
655	07	X X X	•
656	08	X X X	Error location in power form.

TABLE 5.6

5.1.7 Modified Syndromes and Partial Errata Polynomial Table (TMSYN)

- o Table name is TMSYN

- o Purpose

  MSYNG computes the modified syndromes and some product terms of the syndromes and the erasures utilized later in the generation of the errata polynomial.Its output is stored in TMSYN in code form to be available for other subprograms.

- o Size and Indexing

  TMSYN is 17(D) words long (fixed length, vertical type). It is addressed by using the K bus instruction setting to be the data Ram address of the first word of the table. A CPE index register is set to this value and is automatically incremented by one whenever the table is addressed. This provides the address of the next sequential word.

- o Fields

  Table 5.7 describes the fields in detail. Decimal point and scaling are not involved.

5.1.8 Modified Syndromes and Partial Errata Polynomial Table. (TMSYP)

- o Table name is TMSYP

- o Purpose

  The same as table TMSYN. The contents of TMSYP are the same as those of TMSYN except that they are in power form.

- o Size and Indexing

  Indentical to TMSYN

- o FIELDS

  Table 5.8 describes the fields in detail. Decimal points and scaling are not involved.

TABLE NAME: TMSYN

RAM ADDRESS	WORD NUMBER	BIT LOCATIONS 7 6 5 4 3 2 1 0	CONTENTS
561	00	X X X	Number of erasures , s .
562	01	X X X	Errata polynomial terms / Modified syndromes in code form.
563	02	X X X	.
564	03	X X X	.
565	04	X X X	.
566	05	X X X	.
567	06	X X X	.
570	07	X X X	.
571	08	X X X	.
572	09	X X X	.
573	10	X X X	.
574	11	X X X	.
575	12	X X X	.
576	13	X X X	.
577	14	X X X	.
600	15	X X X	.
601	16	X X X	Errata polynomial terms / Modified syndromes in code form.

X = not used

The first s terms are errata polynomial terms . The remaining terms are the

The remaining terms are the modified syndromes.

TABLE 5.7

TABLE NAME: TMSYP

RAM ADDRESS	WORD NUMBER	BIT LOCATIONS 7 6 5 4 3 2 1 0	CONTENTS
602	00	X X X	Number of erasures, s .
603	01	X X X	Errata polynomial terms/ Modified syndromes in power form
604	02	X X X	•
605	03	X X X	•
606	04	X X X	•
607	05	X X X	•
610	06	X X X	•
611	07	X X X	•
612	08	X X X	•
613	09	X X X	•
614	10	X X X	•
615	11	X X X	•
616	12	X X X	•
617	13	X X X	•
620	14	X X X	•
621	15	X X X	•
622	16	X X X	Errata polynomial terms/ Modified syndromes in power form.

X = not used

The first s terms are errata polynomial terms

The remaining terms are the modified syndromes.

TABLE 5.8

5.1.9 Power to Code Conversion Table (TPTCC)

- o Table name is TPTCC

- o Purpose

  Provides the means whereby the conversion from power to code form of Galois field elements is made. Constant values are stored in the table. The values are the code forms of the Galois field elements sequenced by power value.

- o Size and Indexing

  The table is 63(D) words long (fixed length, vertical type). The values are listed twice , except for the last entry, to provide a rapid conversion of powers greater than 31 (these occur after Galois multiplication) into code form,eliminating the need for a subtraction of the value 31 first.

  The table is accessed by using the power value for which the corresponding code is required,as an index. The power value is added to the base address of 400(O) and the resulting address contains the correct code conversion.

- o Fields

  The fields are fixed five bit constants. Reference Figure 5.9

- o Initialization

  Subprogram INITZ copies the constant data from storage in ROM each time the RSED hardware is initiated.

POWER TO CODE CONVERSION TABLE

CPTCC

TPTCC

TPTCC

ROM ADDRESS (D)				CONTENT		RAM ADDRESS (O)	
R	C	P	BITS	BINARY	OCTAL	WORD	BITS
00	08	00	6-2	00000	00	400	4-0
01	08	00	6-2	00010	02	401	4-0
02	08	00	6-2	00100	04	402	4-0
03	08	00	6-2	01000	10	403	4-0
04	08	00	6-2	10000	20	404	4-0
05	08	00	6-2	00101	05	405	4-0
06	08	00	6-2	01010	12	406	4-0
07	08	00	6-2	10100	24	407	4-0
08	08	00	6-2	01101	15	410	4-0
09	08	00	6-2	11010	32	411	4-0
10	08	00	6-2	10001	21	412	4-0
11	08	00	6-2	00111	07	413	4-0
12	08	00	6-2	01110	16	414	4-0
13	08	00	6-2	11100	34	415	4-0
14	08	00	6-2	11101	35	416	4-0
15	08	00	6-2	11111	37	417	4-0
00	08	00	6-2	11011	33	420	4-0
01	09	00	6-2	10011	23	421	4-0
02	09	00	6-2	00011	03	422	4-0
03	09	00	6-2	00110	06	423	4-0
04	09	00	6-2	01100	14	424	4-0
05	09	00	6-2	11000	30	425	4-0
06	09	00	6-2	10101	25	426	4-0
07	09	00	6-2	01111	17	427	4-0
08	09	00	6-2	11110	36	430	4-0
09	09	00	6-2	11001	31	431	4-0
10	09	00	6-2	10111	27	432	4-0
11	09	00	6-2	01011	13	433	4-0
12	09	00	6-2	10110	26	434	4-0
13	09	00	6-2	01001	11	435	4-0
14	09	00	6-2	10010	22	436	4-0
15	09	00	6-2	00001	01	437	4-0

TABLE 5.9 Page 1/2

POWER TO CODE CONVERSION TABLE TPTCC

CPTCC ROM ADDRESS (D)				CONTENT		TPTCC RAM ADDRESS (O)	
R	C	P	BITS	BINARY	OCTAL	WORD	BITS
00	10	00	6-2	00010	02	440	4-0
01	10	00	6-2	00100	04	441	4-0
02	10	00	6-2	01000	10	442	4-0
03	10	00	6-2	10000	20	443	4-0
04	10	00	6-2	00101	05	444	4-0
05	10	00	6-2	01010	12	445	4-0
06	10	00	6-2	10100	24	446	4-0
07	10	00	6-2	01101	15	447	4-0
08	10	00	6-2	11010	32	450	4-0
09	10	00	6-2	10001	21	451	4-0
10	10	00	6-2	00111	07	452	4-0
11	10	00	6-2	01110	16	453	4-0
12	10	00	6-2	11100	34	454	4-0
13	10	00	6-2	11101	35	455	4-0
14	10	00	6-2	11111	37	456	4-0
15	10	00	6-2	11011	33	457	4-0
00	11	00	6-2	10011	23	460	4-0
01	11	00	6-2	00011	03	461	4-0
02	11	00	6-2	00110	06	462	4-0
03	11	00	6-2	01100	14	463	4-0
04	11	00	6-2	11000	30	464	4-0
05	11	00	6-2	10101	25	465	4-0
06	11	00	6-2	01111	17	466	4-0
07	11	00	6-2	11110	36	467	4-0
08	11	00	6-2	11001	31	470	4-0
09	11	00	6-2	10111	27	471	4-0
10	11	00	6-2	01011	13	472	4-0
11	11	00	6-2	10110	26	473	4-0
12	11	00	6-2	01001	11	474	4-0
13	11	00	6-2	10010	22	475	4-0
14	11	00	6-2	00001	01	476	4-0
15	11	00	6-2	00010	02	Not transferred.	

TABLE 5.9 Page 2/2

5.2 Variables

5.2.1 SEPEP

SEPEP is a one word variable located at word 644(O) in the data Ram. All data Ram words are eight bits wide but only bits 4 through 0 are used.

EPVAL stores the address of the error location polynomial in SEPEP for later use by the subprograms. The address may have one of three values 700,711 or 722. Only the lower five bits of the address are stored in SEPEP to provide for an eventual 5 bit data Ram . Later, any subprogram reading SEPEP must supply the most significant bits of the address.

5.2.2 SCBIX

SCBIX is a one word variable located at word 477(O) in the data Ram. All data Ram words are eight bits wide but only bits 4 through 0 are used.

SCBIX is used to indicate the current data block A or B. Any program which uses CPE register R_9 , the normal place for this indicator, for its own use must first store the information in R_9 in SEPEP. Subprograms APGEN and EPGEN set SEPEP.

SEPEP has the value zero when A is the current data block and non zero when B is the current data block.

5.3 Constants

5.3.1 Code to Power and Power to Code Conversion constants

Reference tables 5.4 and 5.9 for a complete definition of these constants.

5.3.2 K Bus

The INTEL 3000 microprocessor has the capability to permit the use of the K bus, its masking bus/register, as a means of providing constants. Each constant used(usually data Ram base addresses) is fully defined in each instruction.

5.4 Flags

5.4.1 Data Read/Write Control Flags.

A full description of these flags is provided in table 5.2.

5.5 Indexes

There are no common indexes used by the subprograms.

5.6 Subprogram Reference

COMMON ITEM	REFERENCING SUBPROGRAMS
IAD07/IBD07	INOUT MSYNG RPVAL
IADCC/IBDCC	EPVAL
IADCS/IBDCS	EPVAL RPVAL
IADEE/IBDEE	APGEN EPVAL RPVAL
IADES/IBDES	APGEN MSYNG EPVAL RPVAL
IADSN/IBDSN	MSYNG
IADXX/IBDXX	INITZ INOUT
IADZY/IBDZY	INITZ INOUT RPVAL
SCBIX	INOUT APGEN EPVAL
SEPEP	EPVAL RPGEN RPVAL
TAPOL	APGEN MSYNG
TEPOL (TEPAA/TEPBB/TEPCC)	INOUT EPGEN EPVAL RPGEN
TEVAL	EPVAL RPVAL
TMSYN	MSYNG EPGEN RPGEN RPVAL
TMSYP	MSYNG EPGEN RPGEN
TCTPC/TPTCC	INITZ APGEN MSYNG EPGEN EPVAL RPGEN RPVAL

5.7 Notes

5.7.1 Subprogram List

The following is a complete list of all subprograms by text name and Mnemonic:

- o APGEN Erasure polynomial generator
- o EPGEN Error polynomial generator
- o EPVAL Error polynomial evaluator
- o INITZ Initialization subprogram
- o INOUT Input/Output control
- o MSYNG Modified syndrome generator
- o RPGEN Errata polynomial generator
- o RPVAL Errata polynomial evaluator